An Architecture Description Viewpoint Wiki based on the Semantic Web Paradigm

Tamburri, D.A.; Lago, P.; Muccini, H.

2010

VU UNIVERSITY - AMSTERDAM

An Architecture Description Viewpoint Wiki based on the Semantic Web Paradigm

by

Damien Andrew Tamburri

A thesis submitted in partial fulfillment for the
Master's Degree in Computer Science - GSEEM Program in Global Software Engineering

in the

Faculty of Exact Sciences
Department of Computer Science - GSEEM Program in Global Software Engineering

September 2010

Declaration of Authorship

I, Damien Andrew Tamburri, declare that this thesis titled, 'Building an Architecture Description Viewpoint Wiki based on the Semantic Web' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a Master's degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Three quarters of the miseries and misunderstandings in the world would finish if people were to put on the shoes of their adversaries and understood their points of view"

Mahatma Gandhi

VU UNIVERSITY - AMSTERDAM

Abstract

Faculty of Exact Sciences
Department of Computer Science - GSEEM Program in Global Software Engineering

Master's Degree in Global Software Engineering and Architecture

by Damien Andrew Tamburri

Today's software architecture practitioners recognize that relevant architectural aspects should be illustrated in multiple views, targeting the various concerns of different stakeholders. Similarly, the research community remarks that architecture descriptions shall be developed to address stakeholders' concerns concentrating on the use of viewpoints for their description. This notwithstanding, we notice today a proliferation of architecture description languages impervious to these guidelines. This imperviousness creates a gap between what practitioners require and what architecture description languages can provide, making it impossible for the former to choose and use the best fit description for his/her concerns.

To fill this gap, this thesis proposes the design and implementation of a Semantic Wiki to gather and relate multiple viewpoints to provide a knowledge-base to leverage the software architects' modeling, decision making and stakeholder communication. The organization, separation and classification of viewpoints provides practitioners with pragmatic information for selecting the most suitable architecture view targeted to specific stakeholder needs, and hence supports them in the architecting process.

Acknowledgements

On a formal note, I wish to thank my mentor Prof. Patricia Lago for her invaluable teachings in the ways of research and academic publishing. Your teachings are amongst the pearls that animate my scientific organum. Thanks to my all-time mentor Prof. Henry Muccini, if it weren't for him I would be probably a computer programmer somewhere around the world, instead of a successful scholar and researcher.

Contents

List of Figures

List of Tables

Thanks To...

Mi sembra doveroso scrivere I ringraziamenti in italiano, non solo per rispetto alla mia natura per diritto di crescita ma anche per permettere a tutte le splendide persone che chiamo amiche di comprendere quanto voglia loro bene e quanto ogni parola qui riportata, ogni passo qui descritto, ogni riga qui catturata, sia dedicata a loro e a quanto bene mi hanno donato, anche solo con il sorriso.

Prima fra tutte la mia guerriera corsara prediletta, Mirella, ti ringrazio per avermi dimostrato che la forza contenuta in un fulmine puo' essere imitata da una semplice persona, con la giusta motivazione ed il giusto slancio - ti saro' eternamente grato.

Ringrazio mio fratello di sangue Albert David, per essersi preso la briga di sopportarmi e darmi consigli - ti voglio bene Broh The Kemist.

Ringrazio i miei genitori, che fra mille peripezie, mille misantropie e mille difetti, hanno il pregio piu' importante - sono i miei genitori... I love you kids, thanks Pap Toto', and thanks Mom Sandy.

Ringrazio i miei fratelli di vita, gli amici, Frank (segnati 110 px e un bacio per lode :), Il Gallo (Mich, dai che ce l'ho quasi fatta, fumaaamm!!), e il Greg per la loro infinita amicizia (e Greg, grazie di essermi venuto a prendere innumerevoli volte a Fiumicino - LLAH!!!), Il Tenente per gli infiniti insegnamenti, Caniello per l'amicizia eccelsa (abbiamo ancora un brevetto in sospeso), Emileo (grande Maestro anch'egli), Pierscuncio (la mia spalla sempre), Luke il Guarins (per tutte le sue imitazioni), Francesco George (la seconda spalla), Ju-Luigi-Tsu (il mio sacro Libro della scienza), Francesco e Gilda*

*(:*******), Vincenzo e Marta del DoC granderrimi (VVBBBB :******), e il resto della truppa di Venafro... Ringrazio Max e Antonella in quel di AQ per la perseveranza nella loro amicizia fraterna :******* - Grazie a tutti ragazzi :)*

Of course, I could never miss in thanking Leonardos for his patience when I invaded his room in amsterdam :) - sign yourself a potion of glory and 100 xp!!!

On a more personal note, I thank myself Damien for having shown unnerving endurance in difficult and superhuman situations, for showing uncanny prowess in handling problems and difficult solutions alike. Thanks for having kept faith in chaos as a supreme force of disequilibrium, thanks for still being crazy, as much as it takes to believe that the world can indeed be changed and still willing to take part of the job. Thanks for not having given up. Thanks Damien, my best friend and first mentor.

E per ultimo, voglio ringraziare tutti quelli che non hanno creduto in me, tutti quelli che mi hanno detto che non avrei avuto successo, che avrei fallito, che sarei caduto, quelli che mi hanno messo bastoni fra le ruote.. Senza di voi, non avrei mai potuto scoprire di quanta forza d'animo posso disporre nello sguardo corsaro che esiste dentro di me. Grazie, perche' nel voler dimostrare a voi che avevate torto, ho avuto tutto cio' che volevo dal cammino che avete costituito davanti a me.

grazie a tutti - non so come andra' in termini di valore accademico, ma voi nel mio cuore, varrete sempre 110 e lode... Damien.

Chapter 1

Introduction

Chapter 1 provides an introduction to all that was done in fulfillment of my Master's degree thesis assignment. Section 1.1 provides a context overview of the problem at hand. Section 1.2 states the problem. Section 1.3 provides the solution we propose, hence deriving a concrete assignment and its goals. Section 1.4 provides the scope within the boundaries of which, the solution will be applicable. Section 1.5 provides the research questions that stem from the problem, these will have to be answered in order for the solution to be accurate and goals be met. Section 1.6 explains the research approach used to tackle the questions posed in order to achieve the solution. Within this section, a design for the proposed solution is present in Section 1.6.4. Finally, section 1.7 provides a roadmap to the remainder of this thesis.

1.1 Setting the Context

The context within which we are operating rotates around four fundamental concepts: *(a)* software architecture; *(b)* software architecture description; *(c)* software architecture description viewpoints; *(d)* software architecture description views. These concepts are intermingled as they gradually clarify of the system they describe, by splitting it into blueprints and atomic blocks which can be understood by system stakeholders. The following introduces each concept in the order of importance we see fit within this effort.
An architecture view as defined by the ISO/IEC 42010, Software and System Engineering - Architecture Description [1] (the internationalized version of IEEE Std 1471 [2], under revision by IEEE and ISO) as "a work product expressing the architecture of a system from the perspective of system concerns" held by stakeholders. An architecture viewpoint, instead, specifies the conventions for constructing a certain view. Furthermore, the key idea of an architecture viewpoint is a directed set of modeling resources

able to address a particular set of system concerns for a particular audience of system stakeholders. All together views, viewpoints and concerns provide a stakeholder-centric perspective of an architecture description, focussing it on specific needs, i.e. stakeholder concerns. The literature on architecture descriptions highlights the need for views to model different concerns of stakeholders, thus addressing the specification of large and challenging architectures [1–8]. However, we currently face two distinct and conflicting trends. On one side, the use of multiple views has become a practice in academia and industry [3, 4, 6, 8]: practitioners in software architecture agreed that adopting separation of concerns and deploying multiple views is the only way to tackle system complexity and therefore project failure [3, 9]. Moreover, practitioners need to understand that, quoting from [3] "[...] Without an architecture that is appropriate for the problem being solved the project will fail. Even with a superb architecture, if it is not well understood and well communicated - in other words, well documented - the project will fail. Not may fail. Will fail". This understanding will necessarily lead IT Architects to adopt multiple viewpoints in order to deploy separation of concerns within software architecture documents, addressing all possible stakeholder concerns [3, 4]. One consequence of the tenet of using multiple views is a growing body of viewpoints that have become available, such as [3, 7, 10–16]. A second consequence is the rise of architecture frameworks as coordinated sets of viewpoints (e.g., Zachman, TOGAF, GERAM, and DODAF4). On the other hand, Architecture Description Languages (ADLs), that are among the most valuable ways to aid the process of constructing and supporting a software architecture [17, 18], are still quite insensible to these needs. While the solution to this problem of misalignment can be seen as our final goal, within the present work we concentrate in the more immediate goal, that of gathering, centralize and render software architecture viewpoints available, since no such mechanism exists to date.

1.2 Problem Statement

A number of mechanisms to support architectural design are present. These mechanisms however produce monolithic models without splitting concerns in multiple views, each covering different stakeholder concerns. This is consequence of the absence of commonly agreed reference for the concise description and selection of views and viewpoints in architecture designs.

1.3 Proposed solution

The proposed solution is to engineer a *Semantic Wiki* [19] for software architecture viewpoints. On one side, a *Wiki* is an online resource repository that allows the creation of any number of interlinked records and has seen much usage in the field of knowledge representation and management [20]. On the other side, *Semantic web* [19, 21, 22] offers technologies to define and support information (a *software ontology* [23]) so that automatic processing of this information is possible.

In our case, a semantic wiki acts as the knowledge manager, whereas ontology engineering [24] mechanisms can codify the way in which viewpoints and their relevant information are made accessible to the community. In order to identify the relevant viewpoint knowledge, our starting point is a set of Viewpoint description templates already available as identified in [2, 25].

1.4 Scope

The scope of this project lies in knowledge engineering. State of the art of semantic wiki technologies - as a representation mechanism for knowledge - is analyzed to produce design alternatives. Ontology engineering is used to realize a viewpoint description ontology. Finally, the chosen semantic wiki technologies use the defined ontology to represent the architecture viewpoints and relate these together.

1.5 Research Questions

In order to engineer a technology matching the previously mentioned expectations, a number of research questions must be investigated:

1. What are the requirements for our semantic wiki?

   The answer to question 1 provides the functionalities and constraints our system must expose; these are the starting point for the design of both the semantic wiki and the underlying ontology itself.

2. What semantic wiki technologies can support us?

   The answer to question 2 provides the data we need to extract design alternatives from the current state of semantic wiki technologies.

3. what ontology engineering method can support us?

   The answer to question 3 contains the qualitative comparison of available ontology engineering practices. Ontology engineering alternatives must be evaluated and a best-fit ontology engineering alternative chosen.

1.6 Research Approach

In order to provide a correct and complete answer to questions 1 through 3 the following research approaches are executed.

1.6.1 Requirements Elicitation

In order to answer question 1, a systematic literature analysis and coding approach is needed. First, publications concerning views and viewpoints must be analyzed and coded to obtain viewpoint specific requirements (What Information we need to support). Secondly, publications concerning semantic wikis must be analyzed and coded to obtain functional requirements. Thirdly, for completeness purposes [26], a Use-Case driven evaluation of our system, must be used to expose additional requirements.

1.6.2 Qualitative Literature Review: Semantic Web Engineering

In order to answer question 2, a state-of-the-art qualitative analysis by means of methods in [27, 28] is carried out. A list of evaluation parameters is developed from the functional requirements. All the technologies in the official W3C semantic web page [1] are evaluated against these parameters. A best-fit semantic wiki alternative is selected and decision rationale is captured, as part of the answer to question 2.

1.6.3 Qualitative Literature Review: Ontology Engineering

In order to answer question 3, a state-of-the-art qualitative analysis by means of methods in [27, 28] is carried out. A list of evaluation parameters is developed from all the requirements which regard the viewpoints' description. An on-line search on the topic of "ontology engineering for semantic web" gives out the candidates to be evaluated - to this list, the official candidates present in the ontology engineering group of interest at W3C is added. The best-fit ontology engineering practice is selected and decision rationale captured, in order to answer question 3.

[1] http://www.w3.org/2001/sw/wiki/Tools

1.6.4 Designing a Solution

Once the preliminary analysis is carried out and the software requirements, scope and involved technologies are well clarified, the software design and implementation begins. The development of the Viewpoint Semantic Wiki proceeds with a prototype driven approach: first we develop a prototype for the architecture and the ontology needed and then we implement both while putting them together. Milestone-based reviews are planned so that the development process involves a fair degree of team interaction. The design solution is contained in chapter 5. The prototype and its implementation together provide the solution.

1.7 Roadmap

The rest of this thesis is divided in 6 chapters. Chapters up through 5 excluded, address the specific research questions aimed at the design solution. The solution itself is contained in chapter 5. Chapter 6 closes this thesis with conclusions and hints to future work.

Chapter 2

Requirements Elicitation: Coding and Use-Case Analysis

This chapter is divided in four Sections: Section 2.1 provides the method we used and the list of requirements we extracted for defining the viewpoint ontology; Section 2.2 provides method and the list of requirements for the functional behavior and structure of the Semantic Wiki; Section 2.3 completes with additional requirements stemming from the analysis of use-cases and existing related technologies, as extracted through the analysis approach in [26]; finally, Section 2.4 discusses results.

The following approaches have been used to reach the results:

Section 2.1 has been realized in two steps: first, a systematic literature review concerning view and viewpoint technologies to gather primary papers; then we coded the information [29] to extract ontology requirements. Here follows the coding approach [29]: search for definitions concerning views and viewpoints; use these definitions as requirements for views or viewpoints; from each requirement, put in evidence the feature that views and viewpoints must exhibit to fulfill.

Section 2.3 has been realized through coding [29] of semantic wiki success stories [30–34] extracted directly from the Semantic Wiki interest group at W3C [1]. Here follows the coding approach [29]: in each story, search for features that make successful each wiki; express each feature in a "may-have" requirement; identify features existing in ALL "successful" semantic wikis and code these as "must-have" requirements.

Section 2.3 has been realized by analyzing existing use-cases [35]. Additional requirements have been extracted through the use of analysis and prototyping method in [26].

[1] http://www.w3.org/2001/sw/

2.1 Ontology Requirements

Since the ISO/IEC 42010 standard for architecture description [2] contains commonly agreed practices for architecture description, we used its definitions for both "view" and "viewpoint" in order to initiate our review. Also, from [2] we extracted keywords related to these and we searched for the combinations:

- *"view"*:
  1. system concern
  2. architecture view
  3. model kind
  4. architecture description
- *"viewpoint"*:
  1. architecture viewpoint
  2. system concern
  3. architecture model
  4. architecture description

We searched for the main keyword by itself and then combined with its relatives, within research papers discovery engines [2] we were able to produce a number of publications concerning viewpoints in architecture description. A single inclusion criterion had to be met: the publication had to concern viewpoints as a primary research topic. The documents we selected as primary studies are: [1–4, 8, 13, 14, 25, 36–41].

Results of the coding are compacted in tables 2.1 and 2.2: column 1 captures a requirement number, which has been subsequently used as a unique identifier for that requirement; column 2 identifies the publication from which the requirement was extracted; column 3 identifies the text of the requirement; column 4 evidences the keyword that was extracted for that requirement.

keywords from column 4 have been used later in Chapter 4 and Chapter 5, as parameters for the evaluation of the ontology engineering approaches and for the implementation of the semantic wiki technology respectively.

[2] ACM Digital Library, DBLP, GOOGLE Scholar, Bibsonomy, IEEE Xplore, SpringerLink

TABLE 2.1: Viewpoint Ontology Requirements

Req. N°	**Pub.**	**Text**	**Extracted Keyword**
VP_1	[36]	*a viewpoint ontology must represent a viewpoint as a loosely coupled and locally managed object.*	Viewpoint Locality
VP_2	[36]	*a viewpoint ontology must encapsulate at least one perspective about the system and domain specified, in a particular representation notation.*	Perspective
VP_3	[41]	*a viewpoint ontology must capture a role and responsibility within the software process.*	Role
VP_4	[41]	*a viewpoint ontology must include the problem statement for the viewpoint.*	Problem Statement
VP_5	[41]	*a viewpoint ontology must capture the exact target point within the software process for which its viewpoint is useful.*	SoftwareProcess Target
VP_6	[36, 41]	*a viewpoint ontology must provide at least these three viewpoint elements:* domain*, delineating the part of the "world" the viewpoint is concerned with;* representation style*, defining the notation used by the specification;* view specification*, expressing the perspective of interest, represented in the style defined;*	Domain, RepresentationStyle, ViewSpecification
VP_7	[36, 41]	*a viewpoint ontology may also carry a* work plan *describing how to build a view, and a* work record *providing a history of the work within the views.*	WorkPlan, WorkRecord
VP_8	[38]	*a viewpoint ontology must provide a viewpoint configuration. it includes the relations with other viewpoints. a viewpoint configuration is a software engineering method as it splits the development process into different viewpoints, each with its own owner.*	Configuration
VP_9	[41]	*a viewpoint ontology must provide a viewpoint configuration owner. the owner is responsible for developing the viewpoint. Viewpoint owners are normally, but not always, human development participants. A non-human Viewpoint owner may be some form of 'intelligent' tool or expert system for example.*	Configuration Owner

TABLE 2.2: Viewpoint Ontology Requirements - Continued

Req. N°	**Pub.**	**Text**	**Extracted Keyword**
VP_{10}	[37]	*a viewpoint ontology must allow viewpoints to interact with a number of other viewpoints and overlap with a number of other viewpoints.*	Viewpoint Interaction
VP_{11}	[37]	*a viewpoint ontology must include inter-viewpoint rules that describe relationships between viewpoints.*	InterViewpoint Rules
VP_{12}	[1, 2, 4]	*a viewpoint ontology must include its target stakeholders.*	Target Stakeholders
VP_{13}	[3, 8, 25]	*a viewpoint ontology must include a software architecture type it is designed to support.*	Architecture Type
VP_{14}	[1, 2]	*a viewpoint ontology must include the concerns it is designed to support.*	Concerns
VP_{15}	[36, 40]	*a viewpoint ontology must include the conventions it uses to construct, interpret and analyze its contents.*	Viewpoint Conventions
VP_{16}	[1, 2]	*a viewpoint ontology may include the definition of one or more model kinds.*	ModelKind
VP_{17}	[36]	*a viewpoint ontology may carry assumptions under which the knowledge and decisions being documented exist.*	Viewpoint Assumptions
VP_{18}	[3]	*a viewpoint ontology must include sources and related work from which it was developed or refined.*	Viewpoint Sources
VP_{19}	[13, 14]	*a viewpoint ontology may carry a meta-model specifying its style.*	Viewpoint Metamodel
VP_{20}	[3]	*a viewpoint ontology must express preferences on the information being captured, i.e. the viewpoint may be used as a specialization of other, more generic, viewpoints.*	Information Preference

2.2 Semantic Wiki Requirements

The publications we analyzed are all those present in the semantic wiki interest page on the W3C home site [3]. Two inclusion criteria were used: *(a)* the publication must reason on the usage of semantic wikis for software knowledge management; *(b)* the publication must present a successful usage of semantic wiki technologies for knowledge management. Using these guidelines we obtained the following publications: [21, 24, 30–34, 42–44].

Results of the coding are contained in table 2.3. Again, column 1 captures a requirement number, which has been subsequently used as a unique identifier for that requirement; column 2 identifies the publication from which the requirement was extracted; column 3 identifies the text of the requirement; column 4 evidences the keyword that was extracted for that requirement.

keywords from column 4 have been used later in Chapter 3, as parameters for the evaluation of the semantic wiki engines.

[3] http://www.w3.org/2001/sw/wiki/Tools

Req. N°	Pub.	Text	Extracted Keyword
SW_1	[34]	*the wiki must provide a visual representation of elements.*	WYSIWYG editor
SW_2	[30, 34]	*the wiki must provide full-text search.*	full-text search
SW_3	[32]	*the wiki must provide mechanisms for on-line, concurrent editing.*	change-tracking
SW_4	[30]	*the wiki must be able to support large client-side data-sets, to cope with increasing size and shape of records*	ACID transactions
SW_5	[30]	*the wiki must provide mechanisms to comment, annotate and review records.*	commenting
SW_6	[21]	*the wiki must enable rating and estimation of popularity for records.*	popularity
SW_7	[30]	*the wiki must enable tracking of editing footprints, i.e. it must be able to keep track of what was contributed and by whom.*	change-tracking
SW_8	[34]	*the wiki must provide semantic search, i.e. it must be able to provide full-text searches within literal property values.*	semantic inference, embedded query, query language
SW_9	[32]	*the wiki must provide context sensitive auto-completion of record searches.*	auto-completion
SW_{10}	[34, 43]	*the wiki must be able to structure the information in different views as well as enabling faceted browsing.*	different views, faceted browsing
SW_{11}	[42]	*the wiki must enable navigation, editing and interoperation of the underlying ontology.*	ontology editor, ontology browser, ontology import, ontology export
SW_{12}	[42, 44]	*the wiki may enable the use of query templates in order to allow advanced information retrieval.*	query templates
SW_{13}	[31]	*the wiki may support the association and visualization of additional data to records (e.g. metamodels, reference publications, graphics etc.).*	different-views, interactive graphical visualization
SW_{14}	[24]	*the wiki must support the definition of cross-reference properties within text.*	semantic inference
SW_{15}	[24]	*the wiki must enable the automated deployment of links between pages.*	context-aware navigation
SW_{16}	[30, 34]	*the wiki must enable typing and annotation of both automatically-defined and user-defined links.*	commenting
SW_{17}	[21]	*the wiki must support different levels of user experience.*	page-rating

TABLE 2.3: Semantic Wiki Requirements

2.3 Requirements from the scenario-driven analysis approach

This Section contains requirements obtained by analyzing semantic wiki use-cases taken from [35] through the methodology in [26]. The use-cases exist in the form of scenarios [35]. Each use-case has been compared with the current requirement list to obtain additional elements. Results are presented in compact form, in table 2.4 the end of this section: Column 1 on this table, again contains a unique identifier for its requirement; Column 2 contains the use-case scenario from which the requirement was extracted; Column 3 contains the text of the requirement while Column 4 contains the keyword associated with that requirement.

2.3.1 Project Coordination Use-Case

Scenario: "Consider a wiki used for coordinating a particular project team within a company. Using semantic technologies, relevant parts of the wiki data shall automatically be gathered by the company's intranet search engine. In the wiki, project members coordinate their activities, and describe their progress on their deliverables. This data can then be collected from the wiki and reused in other applications, e.g. to create monthly report figures, or even up-to-date status reports that are generated on request. As the semantic wiki reuses the company's metadata schema for documents and respects the associated constraints (e.g. no document must have more than one title and topics must stem from a predefined set of topics), the automatic integration into the corporate information infrastructure works smoothly."

1. the wiki ontology must incorporate schema information and constraints from external ontologies for compatibility and reuse.
2. the wiki must be able to track date and time of edits and audits. each edit must be associated to the editing member.
3. the wiki must support or be able to integrate search engines in its local operative environment.

2.3.2 Semantic Wiki Vocabulary Use-Case

Scenario: "assume that an international conference wants to use a wiki for gathering information around the event. Participants can use the system to exchange information about accommodation and travel, to coordinate Birds-of-a-feather (BOF) sessions, or to actively provide links to their presentation material. At the same time, the organizers

publish official schedules on protected wiki pages. Using a semantic wiki, this data can be queried and extended in complex ways, e.g. to provide a scheduling system that suggests sessions and BOF sessions based on a participants interests. Also, if the conference management system supports some form of RDF export, one can initialize the wiki pages with basic information about accepted papers and participants. The ESWC2006 [4] *wiki is based on such a bootstrapped system."*

4. the wiki must refer to existing ontological vocabularies [45, 46] to form its information records.
5. the wiki must provide secured artifacts.
6. the wiki must be able to record files as well as text records.

2.3.3 Semantic Wiki for Personal Knowledge Management Use-Case

Scenario: "the wiki is operated as a desktop application and cooperative editing is not required. Semantic technologies simplify data organization and search, and the machine-processable annotations provide suitable interfaces with other semantic desktop applications. For instance, the wiki can be used to take notes about persons, and one would like to combine this information with address book applications. Using vocabulary from existing ontologies, the wiki becomes compatible with various types of metadata, and thus its information could be used in RDF based desktop tools."

7. the wiki may provide on/off switch for cooperative editing of certain records.
8. the wiki may provide entry points and APIs to interface it with local or remote applications.
9. the wiki may provide ontology as well as record information export.

[4] http://www.eswc2006.org/

TABLE 2.4: Use-Case Based Requirements

Req. N°	**Source**	**Text**	**Extracted Feature**
UCR_1	Project Coordination Use-Case	*the ontology must incorporate schema information and constraints from external ontologies for compatibility and reuse.*	Ontology Import, Ontology Export
UCR_2	Project Coordination Use-Case	*the wiki must be able to track date and time of edits and audits. each edit must be associated to the editing member.*	change-tracking
UCR_3	Project Coordination Use-Case	*the wiki must support or be able to integrate search engines in its local operative environment.*	using query language, full-text search
UCR_4	Semantic Wiki Vocabulary Use-Case	*the ontology must refer to existing ontological vocabularies [45, 46] to form its information records.*	Ontology Import
UCR_5	Semantic Wiki Vocabulary Use-Case	*the wiki must provide secured artifacts.*	-
UCR_6	Semantic Wiki Vocabulary Use-Case	*the wiki must be able to record files as well as text records.*	-
UCR_7	Semantic Wiki for Personal Knowledge Management Use-Case	*the wiki may provide on/off switch for cooperative editing of certain records.*	-
UCR_8	Semantic Wiki for Personal Knowledge Management Use-Case	*the wiki may provide entry points and APIs to interface it with local or remote applications.*	-
UCR_9	Semantic Wiki for Personal Knowledge Management Use-Case	*the wiki may provide ontology as well as record information export.*	Ontology Export

2.4 Discussion of the Results

This Chapter provides two lists of requirements: one for the information that should be stored in a semantic wiki, and one for the types of usages that such semantic wiki should support.

We argue that both lists do represent realistic needs from the community, thanks to our systematic requirements elicitation approach based on a formalized coding methodology. In addition, all requirements are derived from a systematically obtained set of relevant publications in the field. This ensures relevancy.

In discussing the results we obtained, two main facts must be pointed out: *(a)* the articles we analyzed have been selected with a methodical approach; *(b)* the requirements derive from the methodical application of coding to our list of papers.

Fact *(a)* guarantees that the list of publications only contains pertinent and meaningful elements. Fact *(b)* guarantees that the requirements themselves are not "dangling", i.e. they are directly referenced to an "origin" publication and the trace to this "origin" is maintained.

The requirements themselves guide the development of the future chapters. More in particular:

Requirements in tables 2.1 and 2.2 as well the keywords extracted from them, are used in Chapter 4 as parameters to guide the selection of the ontology engineering methodology. These requirements are also used in Chapter 5 to develop the viewpoint ontology and meta-model.

Requirements in table 2.3 as represented by the keywords extracted from them, are used as parameters in Chapter 3 to guide the selection of a semantic wiki engine.

Requirements in table 2.4 and their keywords, are used as additional parameters in Chapter 4 to guide the selection of the ontology engineering methodology.

All the requirements were used in Chapter 5 for implementation. Together, all the requirements captured in tables 2.1, 2.2, 2.3 and 2.4 provide an answer to question 1, namely "*What are the requirements for our semantic wiki?*".

Chapter 3

Qualitative Literature Review: Semantic Web Engineering

Within this chapter we explore Semantic Web Technologies in order to find a best-fit candidate for our needs. We use the requirement keywords from table 2.3 as parameters to guide the selection.

This chapter is divided into two sections: Section 3.1 provides an overview and analysis of each of the technologies that are investigated - each technology is described in a brief summary along with a list of parameters met; Section 3.2 provides the best-fit alternative we have chosen along with the rationale for the choice, according to parameters. Section 3.1 is realized by analyzing documentation on the semantic wiki technologies present in the W3C official semantic wiki technologies list - as available on June the 16th / 2010 [1], the official SemanticWeb interest group for wikis - as available on June the 16th / 2010 [2], as well as a survey specific to the subject [31]. Each semantic wiki technology is provided with a list and description of its parameters met and strongpoints. Section 3.2 is realized by matching each of the alternatives described in section 3.1 with the parameters. The highest scoring technology is selected as a best-fit technology and a rationale is given. The rationale is derived by the parameters met which most match our parameters.

[1] http://www.w3.org/2001/sw/ wiki/Tools
[2] http://semanticweb.org/ wiki/Tools

3.1 Qualitative Literature Review: Semantic Wiki Technologies

This section describes the technologies we evaluated, each presented with its name and publication. For each technology we studied the presenting publication and analyzed tutorials, verifying its compliance with the parameters. The List of parameters met by each technology is below its description. Namely, the technologies are:

1. IkeWiki - [47]
2. Kaukolu - [48]
3. Makna - [49]
4. OntoWiki - [30]
5. PlatypusWiki - [32]
6. Rhizome - [50]
7. SemanticMediaWiki - [51]
8. SemperWiki - [52]
9. SweetWiki - [53]

3.1.1 IkeWiki: a JAVA based semantic Wiki Engine

This engine is a Java Web application that was originally developed as a tool for creating ontologies collaboratively and for managing knowledge. To support users in these tasks, IkeWiki focuses primarily on providing advanced semantic functionalities such as reasoning. This different focus allows IkeWiki to accept lower scalability and higher hardware demands than Semantic MediaWiki. IkeWiki supports developers in using as well as editing OWL ontologies. You can configure it to use OWL-RDF Schema or OWL DL (description logics) reasoning. A rule-based inference mechanism is under development.

List of parameters met:

- full-text search
- inference
- change tracking

- intelligent auto-completion
- ontology editor
- WYSIWYG editor (What You See Is What You Get)
- context-aware navigation
- ontology browser

3.1.2 Kaukolu: a Knowledge-base addressed Wiki Engine

This research prototype is based on JSP-Wiki, an older, very borderline implementation of the wiki concept based on the JSP technology. Kaukolu allows annotations with extended wiki markup as well as form-based annotations that are built dynamically from underlying ontologies. Annotations can refer to arbitrary parts of a page rather than just the whole page, and external systems can generate annotations automatically. For example, experiments are currently under way that use eye-tracking technology and an eye-tracker based extension to highlight text. One application scenario is the annotation of existing documents such as juridical texts.

List of parameters met:

- full-text search
- ontology export
- ontology import
- commenting
- auto completion

3.1.3 Makna: A general Purpose Semantic Wiki Engine

Makna is conceived as a Wiki-based tool for distributed knowledge engineering. It extends an existing Wiki, JSPWiki, engine with generic, easy-to-use ontology-driven components for collaboratively authoring, querying and browsing Semantic Web information. Ontologies are expected to be imported to the Wiki instance by administrators since the system does not include any mechanism to develop, handle and modify ontologies. MaknaWiki users are able to create semantic content (in form of RDF statements referencing pre- configured ontologies) in the classical Wiki manner. They are provided with an extended Wiki syntax and with assistant tools simplifying the interface to the ontologies

employed. Further on, users can create, modify and delete RDF statements associated with Wiki pages.

List of parameters met:

- full-text search
- inference
- query templates
- ontology export
- ontology import
- ACID Transactions
- auto completion
- context-aware navigation

3.1.4 OntoWiki: a semantic web resource for the presentation of knowledge

OntoWiki is a tool providing support for agile, distributed knowledge engineering scenarios. The main goal of the OntoWiki approach is to rapidly simplify the presentation and acquisition of instance data from and for end users. This system differs from the others mentioned in that classical textual content is no longer in the foreground. Instead, OntoWiki offers an easy-to-use interface for collaboratively creating and maintaining ontologies. It also supports semantic search and navigation as well as the possibility of versioning metadata. This technology is one of the most powerful to date, also providing an active community of utilizers behind it. The OntoWiki prototype facilitates different views on record data. To enable users to edit information presented by the OntoWiki system as intuitively as possible, the OntoWiki approach supports online editing as well as view editing.

List of parameters met:

- full-text search
- inference
- change tracking
- commenting

- page popularity rating
- page ranking
- auto completion
- ontology editor
- WYSIWYG editor
- context aware navigation
- different-views visualization
- faceted browsing
- ontology browsing

3.1.5 PlatypusWiki: the semantic wiki wiki web

Platypus Wiki is a project currently under development and refinement. It is available as a prototype of a semantic Wiki Wiki Web which uses RDF models and OWL vocabularies to represent metadata and relations between compliant wiki pages development with it. Platypus Wiki is implemented in Java+JSP and is available as an open-source package using Apache Tomcat as the servlet container. Its underlying semantics divide up all the objects stored in record as concepts, objects or ideas. Each page is grouped under a Topic banner allowing for immediate multiple viewed browsing.

List of parameters met:

- full-text search
- inference
- using query language
- page ranking
- context aware navigation

3.1.6 Rhizome: a wiki like content management and delivery system

Rhizome is by declaration of the authors, an experimental, open source content management framework that can capture and represent informal, human-authored content in a semantically rich manner. It is said to bring about in the wiki information storage, the

new commons of "idea". This commons wouldn't comprise just a web of interlinked pages of content, as is the current World Wide Web, but a web of relationships between the underlying ideas and distinctions that the content implies: a permanent, universally accessible interlinking of content, based on imputed semantics such as concepts, definitions, or structured argumentation.

List of parameters met:

- full-text search
- using query language
- ontology export
- change tracking
- commenting
- different views visualization

3.1.7 SemanticMediaWiki: powering wikipedia's wiki with semantic web

This semantic wiki engine focuses on the Wikipedia-encyclopedia scenario and emphasizes on scalability and backward compatibility. It has no predefined schema and no ontology is required for annotations, so users can add new annotations as needed - similar to tagging systems. Because efficient and freely available inference systems that scale up to the size of Wikipedia aren't foreseeable in the near future, Semantic MediaWiki doesn't support inferencing and similar advanced functionalities.

List of parameters met:

- embedded query
- full-text search
- using query language
- ontology export
- ontology import
- change tracking
- ACID Transactions
- context aware navigation

3.1.8 SemperWiki: a semantic personal wiki

SemperWiki stems from the idea to allow personal data to include semantic aspects within it. It is mainly intended as a system to handle modify and maintain personal information. Its main focuses are ease of use and user-friendliness. SemperWiki is not limited to the annotation of desktop data; it is a general-purpose tool for creating and using semantically annotated data that addresses a basic prerequisite towards a better desktop: helping users to add semantic annotations. SemperWiki stores all information in RDF. The collection of triples that SemperWiki stores form a valid RDF model and can directly be exchanged with others. For the semantic annotations it uses a very simple syntax. A statement is written on a line by itself and consists of a predicate followed by an object. Such a statement is expanded to a triple using the URI of the page as a subject.

List of parameters met:

- full-text search
- using query language
- context aware navigation

3.1.9 SweetWiki: Semantic WEb Enabled Technology

This research prototype from Inria Sophia-Antipolis is implemented in Java. SweetWiki combines social tagging with formal ontologies. Users can easily annotate pages with arbitrary tags, which they can in turn associate with concepts from the underlying ontologies. In addition, SweetWiki uses the Corese [54] inference machine, which was developed for conceptual graphs and offers many reasoning services. This technology is particularly difficult to use given its formal development logics.

List of parameters met:

- full-text search
- inference
- auto-completion
- ontology editor
- WYSIWYG editor

- context aware navigation
- faceted browsing

3.2 Qualitative Literature Review: Best-Fit Candidate Selection

In order to choose a best-fit candidate we considered the count of parameters met by each wiki. Since each parameter is weighted equally, the best-fit is the technology which meets the most parameters. The rest of this section provides a comparison of the technologies, before the final selection is made.

Contrarily to all of the other technologies analyzed (with the exception of OntoWiki and SWEETwiki) IkeWiki provides an ontology editor. Similarly to most others, it allows browsing of the ontology itself. Mechanisms to import or export the ontologies themselves, are missing. While it shares a lot of similarities with a complete technology such as OntoWiki, it misses commenting and interoperability facilities.

Kaukolu's stage of development seems too embryonal for it to be used effectively without external integration. Contrarily to most of the other wikis, it does not offer any editing or browsing of ontologies, which means that ontology development takes place somewhere else and is later imported into it. This shortcoming hinders general usability.

Makna provides ACID transactions contrarily to most of its brothers, with the exception of SemanticMediaWiki. ACID Transactions guarantee reliability of DB transactions, which is very valuable on sensible record content. Moreover, contrarily to most others, Makna provides editable and compilable source code. Code is based on JSPwiki which enhances potential for further development and interoperability.

OntoWiki is one of a handful that provide page ranking and popularity rating which can allow reasoning on records' effective value. Second, OntoWiki provides an ontology editor in the form of a WYSIWYG editor which allows easy and agile editing of records and ontology as well. Last but not least, OntoWiki provides different views visualization which is present only in Rhizome. On the other hand, it misses ACID transactions contrarily to some technologies we have analyzed such as Makna and SemanticMediaWiki.

What was said for Kaukolu is equally valid for Platypus Wiki. It is still an embryonal technology although its strongpoints are promising. Unlike all the technologies we have analyzed to this point, Patypus provides the possibility to query for records with a complex query language. Differently from most of the others, it does not provide any mechanism to handle ontologies whatsoever.

	IkeWiki	Kaukolu	Makna	OntoWiki	PlatypusWiki	Rhizome	SMediaWiki	SemperWiki	SweetWiki
Authoring									
ACID Transactions	○	○	●	○	○	○	●	○	○
auto-completion	●	●	●	●	○	○	○	○	●
ontology editor	●	○	○	●	○	○	○	○	●
WYSIWYG editor	●	○	○	●	○	○	○	○	●
Navigation									
context-aware navigation	●	○	●	●	●	○	●	●	●
different views	○	○	○	●	○	●	○	○	○
faceted browsing	○	○	○	●	○	○	○	○	●
interactive graph visualization	○	○	○	○	○	○	○	○	○
ontology browser	●	○	○	●	○	○	○	○	○
Retrieval									
embedded query	○	○	○	○	○	○	●	○	○
full-text search	●	●	●	●	●	●	●	●	●
semantic inference	●	○	●	●	●	○	○	○	●
query templates	○	○	●	○	○	○	○	○	○
using query language	○	○	○	○	●	●	●	●	○
Reuse									
Ontology import	○	●	●	○	○	●	●	○	○
Ontology export	○	●	●	○	○	○	●	○	○
Social Collaboration									
change-tracking	●	○	○	●	○	●	●	○	○
commenting	○	●	○	●	○	●	○	○	○
popularity	○	○	○	●	○	○	○	○	○
page-rating	○	○	○	●	●	○	○	○	○
Parameters Met	**8**	**5**	**8**	**13**	**5**	**6**	**8**	**3**	**7**

TABLE 3.1: Overview and Comparison of parameters met for the Wiki Engines.

Rhizome shows a number of valuable points such as using a query language to access stored records as well as providing change tracking facilities. These two feats are combined with the possibility to visualize both queries and simple records in different views.

SemanticMediaWiki is the father to OntoWiki. Its core technology was used by the OntoWiki engineers as a starting point to design OntoWiki itself. Therefore, both OntoWiki and SemanticMediaWiki share parameters as ontology handling, embedded queries, Context aware navigation and track of changes on records as well as identification of changes. On the other hand, a proper ontology editor is missing in SemanticMediaWiki.

SemperWiki with its simplistic view of semantic wikis deprives itself of a number of parameters met such as ACID transactions or ontology handling. User friendliness is therefore bought at the expense of usability. It is nowhere near the support we need.

SweetWiki is the only technology which provides advanced and formal ontology editing. It figures as an essential tool to develop formal knowledge management wikis. It allows agile and formal ontology editing, but provides little support and low user friendliness. It can be seen as the exact opposite extreme to SemperWiki. Both technologies are therefore unfeasible as supporting technologies.

Table 3.1 summarizes the investigation results. By comparing the technologies at hand and looking for the one which maximized the parameters met we chose as best-fit candidate OntoWiki. This technology maximizes the matches in table 3.1 - this technology is selected as our source platform. Going back to our research questions, the investigation and literature review into Semantic Wikis and the selection of a best-fit with its rationale answer question 3, namely "*What semantic wiki technologies can support us?*".

Chapter 4

Qualitative Literature Review: Ontology Engineering

This chapter is divided into three sections: Section 4.1 provides the parameters used for the evaluation; Section 4.2 explains the ontological engineering technologies we considered and their evaluation; Section 4.3 explains the best-fit candidate and the rationale for the choice.

Section 4.1 is realized by scanning all the requirements concerning the expected functional characteristics of our ontology, more specifically the requirements regarding the viewpoint ontology. From these requirements a list of desirable ontology engineering features is extracted. These are subsequently matched against the candidates identified in section 4.2. Section 4.2 is realized by gathering all ontology engineering technologies present in the official Semantic Web ontology engineering page [1]. To these are added all technologies resulting from an on-line search on the topic of "ontology engineering for semantic web". Each technology is first analyzed briefly; then, a comparison is made with other candidates; finally, a list of features is provided. In section 4.3 the list of features is matched against the parameters from section 4.1. Section 4.3 is realized by checking each candidate from section 4.2 through the parameters identified in section 4.1. The highest matching technology is selected as best candidate. The rationale further explains the choice by analyzing the parameter-matching features in the selected technology.

[1] http://semanticweb.org/wiki/ Ontology_Engineering

4.1 Qualitative Literature Review: Ontology Engineering - Parameters

This section introduces the list of parameters that the ontology engineering must meet according to requirements. The parameters are extracted either from the requirements in brackets or from previous reported experience [55]. Note that the requirements in brackets are represented through their requirements number, as assigned in tables 2.1 and 2.2.

1. ontology faceted editing and viewing (Requirements VP_6 and VP_7).
2. ontology commenting (Requirement VP_7).
3. ontology edit tracking (Requirement VP_6).
4. ontology auto-documenting (Requirements VP_4, VP_5 and VP_6).
5. ontology modeling (Requirements VP_{16}).
6. ontology cross-platform compatibility [55].
7. WYSIWYG ontology editor [55].

4.2 Qualitative Literature Review: Ontology Engineering - Candidates

This section describes the technology under evaluation. The candidates were extracted from the official Semantic Web ontology engineering page and integrated by results from an on-line search on the topic of "ontology engineering for semantic web". The list of technologies presents the name of the technology and the main paper exposing it. Note that for those technologies that do not provide an editor, the OntoWiki editor can be used.

1. DILIGENT [56]
2. OTK methodology [57]
3. METHONTOLOGY [58]
4. Ontology Development 101 [59]
5. Uschold / King [60]

6. HCOME [61]
7. DOGMA [42]
8. UPON [62]
9. DKAP IDEF5 [63]
10. CO4 [64]
11. BioInformaticSemantics [65]

Note that from all these results, the technologies deemed irrelevant to our domain because of their different target domain (BioInformaticSemantics [65], CO4 [64] and Uschold / King [60]) are only investigated to verify their effective domain incompatibility: BioInformaticSemantics was disqualified since its main domain is that of Bioinformatics; CO4 was disqualified since its domain is that networked repositories of knowledge; Uschold / King was disqualified since its domain is that of Enterprise Engineering. The rest of the chapter investigates the remaining candidates.

4.2.1 DILIGENT

DILIGENT is a joint effort from the university of Lisbona and the university of Karlsruhe. The DILIGENT methodology, is by declaration of the authors, " [...] intended to support domain experts in a truly distributed setting to engineer and evolve ontologies with the help of a fine- grained methodological approach based on Rhetorical Structure Theory (RST) [66]". It is intended to propose a model for knowledge engineering which is agile, distributed and continuously evolving. The DILIGENT process starts by building a draft ontology by the hands of domain experts. An adaptation step successfully tailors the draft to the scenario at hand and then an analysis of the gaps this tailoring has with the actual problem (from the ontology user's perspective). Last two steps envision continuous evolution and local rework of the ontology itself by means of team meetings. Within DILIGENT, the Rhetorical Structure Theory is used as the theoretical basis for the initial construction step and the later reworking steps. DILIGENT does not provide its own editing environment.

List of features:

- ontology commenting, since the methodology envisions collaborative development.
- ontology edit-tracking, since the methodology is based on continuous revision of local ontology updates.

- ontology modeling , since the methodology envisions RST as the theoretical underpinning of its structure.

4.2.2 OTK methodology

OTK (On-To-Knowledge) methodology is a joint effort of a number of Universities (Vrije University - Amsterdam, University of Karlsruhe) and a number of industrial parties (Swiss Life - Switzerland, CognIT, etc.). OTK was designed as an effort to support all knowledge handling tasks possible, from extraction to search to maintenance. OTK presents as a strong project with valuable verifications and validations. IT provides a visual tool to develop and analyze ontologies and also provides a thorough documentation of its structure and main applications.

List of features:

- ontology commenting, through the built-in tool suite
- ontology edit-tracking, through the built-in tool suite
- ontology auto-commenting, through the automatic inference capabilities of its tools
- ontology modeling, given its UML-like superstructure
- WYSIWYG editor, given the presence of an extensive tool suite

4.2.3 METHONTOLOGY

METHONTOLOGY is the fruit of the Artificial Intelligence innovation group within the University of Madrid. METHONTOLOGY is based on the paradigm of prototyping. The base idea is that of developing an initial ontology draft (prototype) and evolving this prototype. The prototype itself is built through knowledge harvesting from the domain, and conceptualization of "important" concepts. The selection of the "importance" criteria is also aided by the methodology. Implementation of the ontology thus designed, comes through the use of previously existing formal languages such as CLASSIC, BACK, OntoLingua or Prolog. It should be noted that the specification and design processes are very well structured. Once again, a properly designed tool-support is missing.

List of features:

- ontology modeling, through the use of dedicated diagrams for knowledge representation

- ontology commenting, since the methodology envisions the evaluation through proper documentation and commenting of all the prototypes developed and the steps of refinement undertaken

4.2.4 Ontology Development 101

101 is a methodology to build ontologies incrementally and from a very low level of understanding of ontologies themselves on the modeler side. 101 is an effort from the University of Stanford. Once again this technology does not offer an editor on its own but rather uses ProtegŔ-2000. The methodology is based on a series of steps: the outcome of each is a diagram and a set of documents describing the step, the procedure taken the decisions made and so on. Indeed 101 is a valuable methodology to approach the task of ontology engineering as a first time ontologist.

List of Features:

- ontology modeling, through guided usage of external tools
- ontology commenting, through guided documentation and collaborative development

4.2.5 HCOME

This effort is the fruit of the Artificial Intelligence group at the University of the Aegean - Greece. Essentially the approach rotates around a human-centered approach to the engineering of ontologies and knowledge maintaining mechanisms. A strong accentuation is posed in the active developer and user of the ontological technology and ontologies themselves are developed, supported and maintained according to the knowledge of such actors' abilities. The adaptability of the ontology itself rotates around the "knowing" process of the workers themselves. Personal conceptualization is not abhorred in the approach, rather, it is taken into full account. Human Centered Computing is the guiding paradigm underneath the technology. HCOME essentially envisions personal conceptualization followed by group agreement and evaluation as the key steps to any serious ontology engineering attempt. The HCONE tool is designed as a prototype to manage and support the HCOME process.

List of features:

- ontology commenting, since its process envisions sharing of conceptualizations - with opinions and properties to be specified on the conceptualizations themselves

- ontology edit-tracking, since the process envisions the sharing and evaluation of different versions of the conceptualizations first and the ontologies later.
- WYSIWYG editor, given the presence of a tool-suite prototype.
- ontology modeling, given the HCONE tool's tree representation for ontologies.

4.2.6 DOGMA

This technology was designed at the Brussels Vrije University - STARLabs. The methodology is specifically designed for the engineering of formal ontologies targeted at handling persistence data layers (knowledge entities). Several key issues, such as knowledge reusability and shareability are addressed within the technology. The DOGMA approach divides up an ontology development into two main phases: (a) the development of the ontology base, a set of context-specific binary fact types which called "lexons" and (b) instances of their explicit ontological commitments, i.e. the concepts these want to represent. The methodology is well-proven and remarkably formal sharp in definition. The approach provides the DOGMAModeler that supports the whole development process.

List of features:

- WYSIWYG editor, given the presence of its own tool-suite.
- ontology modeling, given the presence of the DOGMAModeler.
- ontology edit-tracking, since the DOGMAModeler is set atop the DOGMA Server infrastructure which stores and serves up the ontology being developed.
- ontology commenting, since the commit system is comment-able.
- ontology auto-documenting, since the technology implementing DOGMA is JAVA driven and hence, auto-documenting is available.

4.2.7 UPON

UPON is an interesting attempt based on the idea of building ontologies with the Unified Development Process. Modeling comes via the UML technology. UPON is use-case driven, iterative and incremental in process structure. UPON is use-case driven in that it aims at producing an ontology with the purpose of serving its users, both humans and automated systems. The nature of the process is iterative because each activity is repeated possibly concentrating on different parts of the ontology being developed, but also incremental, since at each cycle the ontology is further detailed and extended. During

each iteration, five workflows take place: requirements, analysis, design, implementation and test. The approach is very interesting and powerful in nature but it cannot express its full potential given its loose tool support.

List of features:

- ontology modeling, via UML.
- ontology commenting, thanks to the intrinsic mechanisms of UML and its numerous tool suites.

4.2.8 DKAP IDEF5

This research develops a methodology called Domain Knowledge Acquisition Process (DKAP) for creating an ontology of product and process design using IDEF5 [2] and generates a consistency matrix for checking the accuracy of captured information. DKAP rotates around the concept that harvesting knowledge information is as much important a step as representing it. The harvesting, organization and recollection of information is as much important as the use of the information itself. Unfortunately again, the methodology does not provide a proper tool support.

List of features:

- ontology commenting, since the document templates provided envision strong commenting

4.3 Qualitative Literature Review: Ontology Engineering - Best Fit

To choose a best-fit candidate we counted the parameters met from (section 4.1). The rest of this section provides a comparison of the technologies, a visualization of the results in compact form (cfr. table 4.1) and the decision made for the best-fit candidate.

DILIGENT, unlike most of the other approaches investigated, uses formal technologies to draw conceptualizations and hence to draft an initial ontology. Moreover, unlike most of the other formal approaches, DILIGENT maintains agile. Its incremental approach is only comparable to 101. Unfortunately, like most of the other technologies investigated it lacks proper tool support, hence making it difficult to harness its full potential.

[2] http://www.idef.com/IDEF5.htm

	DILIGENT	OTK	METHONTO-LOGY	101	HCOME	DOGMA	UPON	DKAP
Faceted editing and viewing	○	○	○	○	○	○	○	○
commenting	●	●	●	●	●	●	●	●
edit-tracking	●	●	○	○	●	●	○	○
auto-documenting	○	●	○	○	○	●	○	○
modeling	●	●	●	●	●	●	●	○
cross-platform	○	○	○	○	○	○	○	○
WYSIWYG editor	○	●	○	○	●	●	○	○
TOTAL	**3**	**5**	**2**	**2**	**4**	**5**	**2**	**1**

TABLE 4.1: Ontology Engineering Methodologies - Overview

The main strength of the OTK approach is its origin and background in industrial settings. This makes it almost uniquely valid in any ontology engineering attempt. Indeed it also provides a valid tool support alike only a few of the other technologies (e.g. HCOME or DOGMA). OTK is also uniquely provided with an extensive documentation, while most of the others provide little actual detail and success stories of the technologies they describe. OTK is indeed a very good candidate for adoption.

METHONTOLOGY is a valuable effort confronted with its kin since it provides prototyping. This methodology is able to rapidly provide a draft of a working ontology and is actually able to deliver a semi-formal model of it. Again, like DILIGENT, it does not provide a proper tool and therefore is not fully exploitable.

101 is another incremental approach to ontology engineering like OTK. Unlike OTK , 101 does not offer a proper tool to support it. On the other hand, 101 is perhaps the most effective approach to be adopted by the beginning ontologist, since its approach is easily comprehendible and very well documented.

HCOME is a peculiar approach, different from all that was investigated to this point, since it poses much more stress on the human factor involved in ontology engineering. The Human Factor, as it is called within the technology is taken into account while developing the ontology and while using it and supporting it. No other technology encountered poses this stress on the Human Factor. HCOME also provides a valuable tool and therefore it is indeed a valuable candidate for selection. It would seem however that, since our environment uses ontologies as an underlying mechanism rather than an effective description technology, the human factor assumes in our case a secondary importance.

DOGMA is perhaps the most fit technology for our case since, unlike most of its brothers, it offers mechanisms to tackle serious problems such as reusability and cross-platform compatibility. DOGMA also offers a proper tool to support its peculiarly structured process. One more reason for DOGMA to be actually adopted is that it specifically targets, persistence layers of data. DOGMA should be carefully considered for these peculiarities.

UPON seemed the newest approach investigated, given its support to RUP and UML. So new, unfortunately, that it lacks proper tool support. While most of its kin provide some loose specification of tool support or some reference, UPON unfortunately does not. Therefore it should be discarded.

DKAP is an attempt at specifying a formal and universal ontology engineering method. DKAP provides formal proofs and formal mechanisms to verify the ontologies being developed. DKAP however, does not offer tool support.

Table 4.1 presents our analysis results in compact form. As it can be seen from the table, OTK and DOGMA both seem equally fit to develop our ontology. We opted for DOGMA given its possibility to model and store the ontology into a DB which can be used by the semantic wiki engine itself. Going back to our original research questions, the choice of DOGMA with its rationale answers question 4, namely "*what ontology engineering methods can support us?*".

Chapter 5

Solution Design: Semantic Wiki for Architecture Viewpoints - VPWiki

This chapter is divided into two sections: Section 5.1 presents a prototype of the Viewpoint Semantic Wiki (VPWiki); Section 5.2 provides models and implementation of the VPWiki prototype defined in 5.1.

Section 5.1 is realized by fulfilling the requirements from Tables 2.1 and 2.2 as well as 2.3 and 2.4 with an ontology and architecture prototypes respectively. Section 5.2 is realized by implementing the architecture prototype from Section 5.1 as well as implementing the ontology defined in Section 5.1 with DOGMAmodeler and importing it within VPWiki.

5.1 VPWiki Prototype

The requirements we have gathered concern both the semantic wiki and its underlying ontology: from requirements in Tables 2.1 and 2.2, a vocabulary and meta-model must be produced.

Moreover, from requirements in Tables 2.3 and 2.4 an architecture prototype for the semantic wiki can be drawn.

The meta-model for the ontology is captured in figure 5.5 while the architectural prototype for the semantic wiki is captured in figure 5.1. Finally, the vocabulary for our ontology is found on table 5.1.

5.1.1 VPWiki Architecture Prototype

The prototype architecture is depicted in figure 5.1.

The elements in figure 5.1 represent the server-side semantic wiki architecture. Its components are explained as follows:

- *Visualizer*: The visualizer component implements the visualization requirements of our wiki. It implements the simple or semantic full-text search and the querying of the DB by means of advanced features such as query templates. In this component resides the handler for commenting and advanced annotations of records and pages. The visualizer component uses the editor component to enable users to add, remove or edit a record directly from the browser. This component is be instantiated by the main controller component "ServingManager" and it uses it to store and retrieve data.

- *Editor*: The editor component provides all the logic that enables editing of records and creation of new ones. Editing of the ontology is also activated within this component. The facilities to rate and rank pages is resident here. This component is instantiated by the main controller component "ServingManager" and it uses it to store and retrieve data.

- *ServingManager*: The serving manager component acts as the controller element in the architecture. Its subcomponents: *(i)* animate the ontology; *(ii)* store and retrieve data interacting with the storage component; *(iii)* and act as a search engine within the Storage component. The "ServingManager" component is invoked as a main web server entity.

- *Storage*: The storage component handles persistence of data. Transactions are redundant, safe and secure. Failsafe mechanisms are also envisioned.

In addition to these, two general constraints are applied: *(i)* the wiki must allow different views to be visualized in faceted browsing (this constraint is applied to the Visualizer component); *(ii)* the viewer must be able to provide visualization of additional data saved along with records (i.e. models and meta-models, links, publications, documents etc.).

The general pattern of the architecture follows the MVC structure. Essentially, it is a thin client - thick server web architecture. Both the ontology and the architecture are reworked and retouched along the rest of this work.

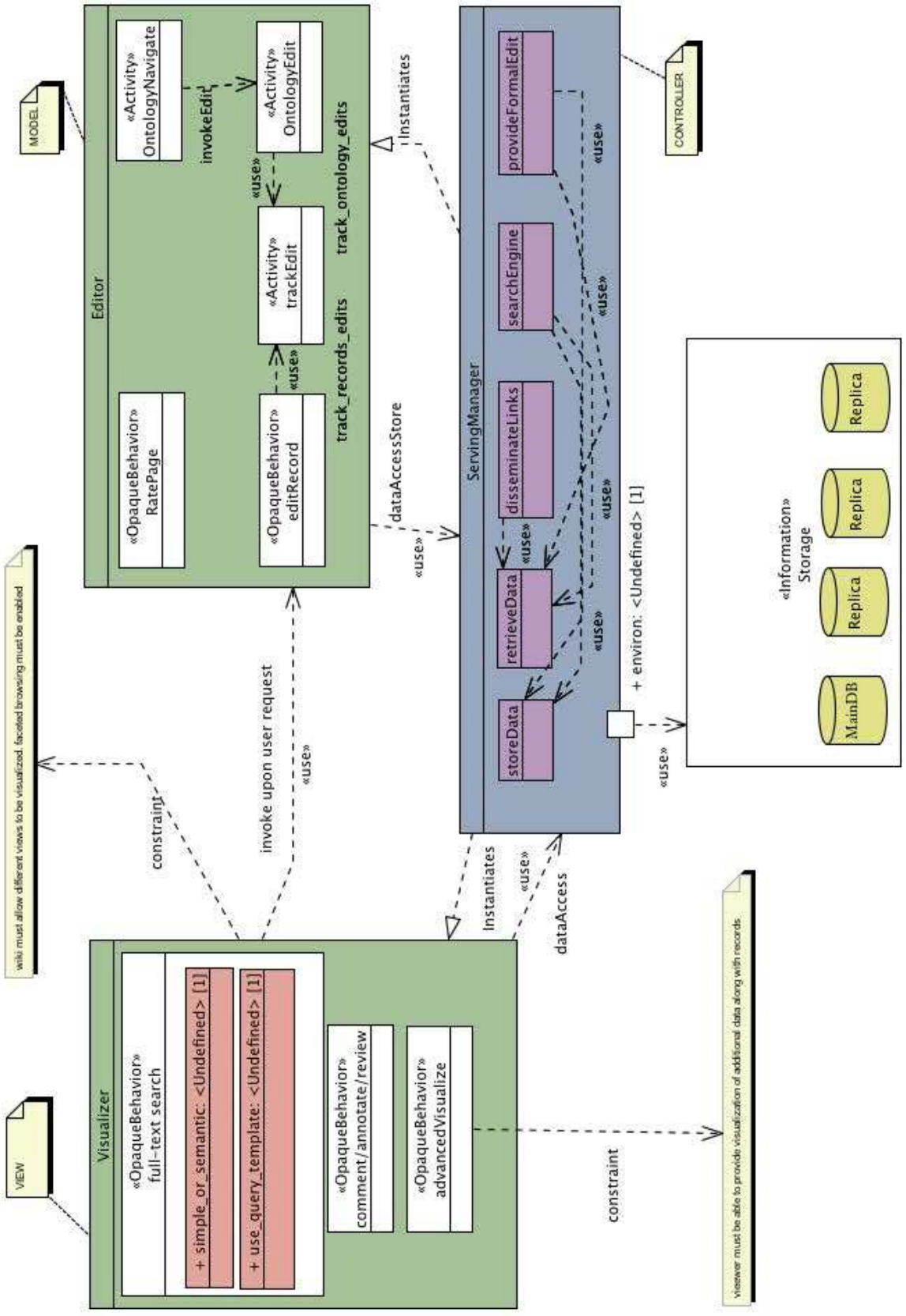

FIGURE 5.1: VPWiki: a High Level Architecture.

Viewpoint	ArchitectureView	Architecture	Stakeholder
ViewpointLocality	RepresentationStyle	System-of-interest	SystemConcern
Stakeholder	ArchitectureModel	ArchitectureDescription	System-of-interest
SystemConcern	ArchitectureViewpoint		ArchitectureDescription
ModelKind	SystemConcern		ArchitectureViewpoint
ArchitectureView			
ViewpointSource			
Domain			
ArchitectureDescription			
Configuration			
RepresentationStyle			
ViewpointRole			

TABLE 5.1: Semantic Viewpoint Wiki: Ontology Vocabulary

5.1.2 VPWiki Ontology Prototype

The ontology prototype is made up of two parts: the viewpoint ontology *vocabulary* contains the lemmas (i.e. concepts) present within the viewpoint ontology; the viewpoint ontology *meta-model* puts in evidence the relations between every lemma and the others.

To construct a minimal yet complete vocabulary, we took as lemmas all the concepts defined in the ISO / IEC 42010 standard for architecture description [2], since it is by definition a minimal set of elements and massively viewpoint-oriented at the same time. Then we added to these elements, all non-overlapping (i.e. with different meaning) keywords from requirement tables 2.1 and 2.2. This process is summarized in Figure 5.2. The viewpoint ontology vocabulary resulting from this process is shown on table 5.1.

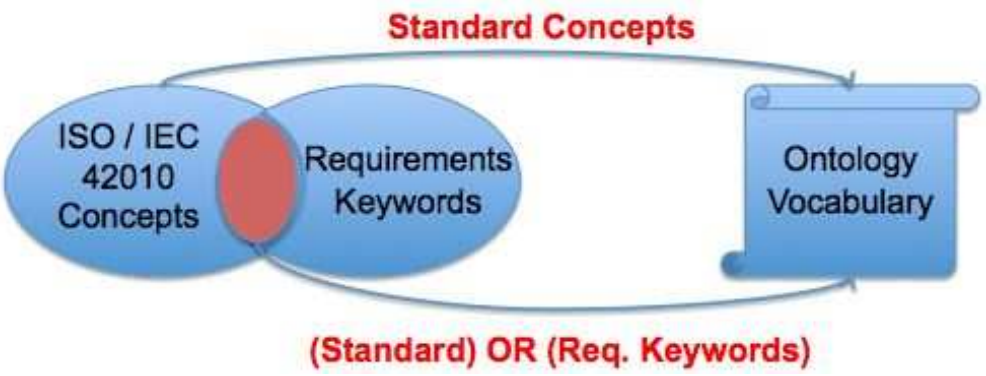

FIGURE 5.2: Viewpoint Ontology Vocabulary Generation.

To construct the meta-model we followed an approach similar to the construction of the ontology vocabulary. We took the core model from the ISO / IEC 42010 Standard [2] (which is infact a meta-model) and augmented it with all keywords from the requirements. If a keyword was dangling (i.e. if the requirement text showed no relation with any of

the others or any element of the ISO / IEC 42010 core model) it was dropped and the rationale for this, captured. This process is summarized in Figure 5.3.

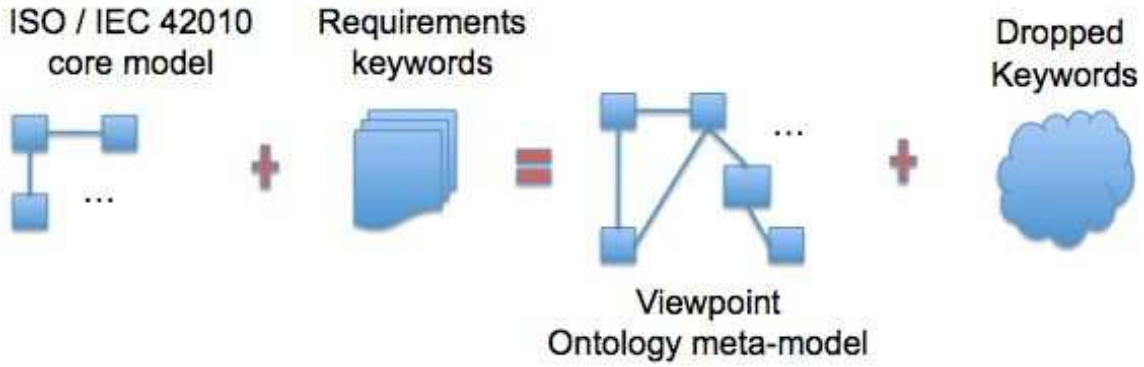

FIGURE 5.3: Viewpoint Ontology Meta-model Construction.

The core component model from ISO / IEC 42010 [2], our starting point, is shown in Figure 5.4.

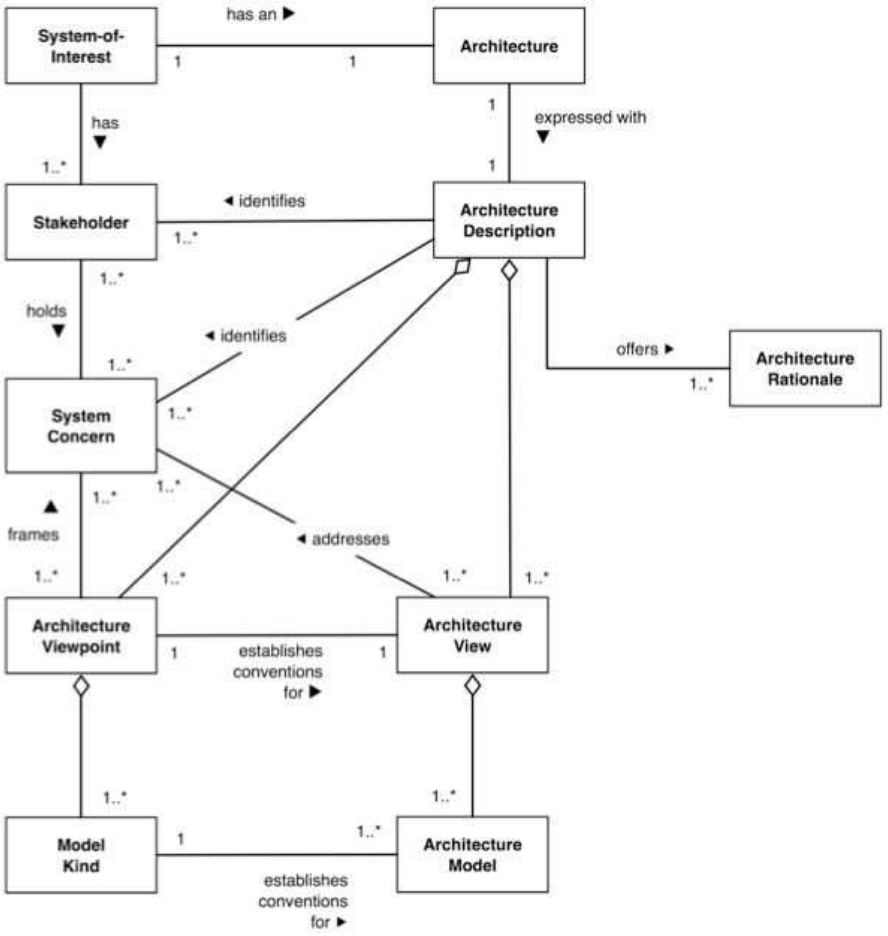

FIGURE 5.4: The ISO / IEC Core Model.

we augmented this model with all requirements keywords on tables 2.1 and 2.2. In order to keep the ontology meta-model both general and minimal, hence following the same philosophy of the ISO / IEC 42010 standard [2], we dropped all the keywords which specialized other keywords. Now follows the full list of keywords, in bold the ones used to decorate the core model in Figure 5.4, in italic the dropped ones:

- **ViewpointLocality**: this is a property of viewpoints and is to be added as a property within the viewpoint entity. This keyword is therefore not dropped.
- **ViewpointPerspective**: this keyword is a meta-class to be added. It represents a set of description rules which can be applied to viewpoints, when these are describing a particular domain or system of interest. According to text, perspective is related to domain and representation style.
- **ViewpointRole**: this keyword is a meta-class to be added. It represents the role a viewpoint is designed to accomplish in a viewpoint configuration. According to text, the viewpoint role is related to viewpoint.
- *ProblemStatement*: this keyword is dropped since its meaning overlaps with system-of-interest from ISO / IEC 42010.
- *SoftwareProcessTarget*: this keyword is dropped since its a specialization of ViewpointRole.
- **Domain**: this keyword is a meta-class to be added. It represents the domain towards which the viewpoint is directed. According to text, this meta-class is related to viewpoint.
- **RepresentationStyle**: this is a meta-class to be added. It represents the style with which the views for a viewpoint should be described. According to text, the RepresentationStyle is related to both view and viewpoint.
- *ViewSpecification*: this keyword is dropped since its meaning overlaps with that of RepresentationStyle.
- *WorkPlan*: this keyword is dropped since it's a specialization of the RepresentationStyle keyword.
- *WorkRecord*: this keyword is dropped since it's a specialization of the RepresentationStyle keyword.
- **Configuration**: this is a meta-class to be added. It represents a set of viewpoints collaborating together to explain a number of concerns within the system. According to text, this meta-class is related to Viewpoint and Concern.

- **ConfigurationOwner**: this is a property of Configuration and must be added within the meta-model accordingly. This keyword is not dropped.
- *ViewpointInteraction*: this keyword is dropped since it's a specialization of the Configuration keyword.
- *InterViewpointRule*: these rules are the constituting element of the Configuration meta-class. They are there therefore a specialization of the Configuration meta-class itself. This keyword is therefore dropped.
- **TargetStakeholder**: this keyword is added as a relation within the ISO / IEC 42010 core model. It is not dropped as it represents the stakeholder(s) to which the viewpoint is targeted.
- *ArchitectureType*: this keyword is dropped since its meaning overlaps with that of the meta-class Architecture.
- *Concern*: this keyword is dropped since its meaning overlaps with that of the meta-class SystemConcern.
- *ViewpointConventions*: this keyword is dropped as it specializes the meta-class RepresentationStyle.
- *ModelKind*: this keyword is dropped since it is already present in the core model from [2].
- *ViewpointAssumptions*: this keyword is dropped since it specializes the Domain meta-class.
- **ViewpointSource(s)**: this is a property of viewpoints. This keyword must be added as such in the meta-model.
- *ViewpointMetamodel*: this keyword is dropped since it specializes the RepresentationStyle meta-class.
- *InformationPreference*: this keyword is dropped since it specializes the meta-class RepresentationStyle.

By adding the selected keywords and properties from the above list we obtained our final meta-model, presented in Figure 5.5. Figure 5.6 provides a tracing of the requirements on the meta-model. More specifically, the figure shows what requirement was used to introduce elements on the meta-model: with reference to tables 2.1 and 2.2, each requirement is identified with its ID in the form: VP_x, where "x" is the number of the requirement in the respective tables.

It should be noted that reproducing the relations from the requirements' text onto the meta-model is an exercise of interpretation, since the relations themselves are not formalized in any way but loosely specified and therefore arguable. But given the formal process with which these relations were obtained, we argue that these are expressing valuable and meaningful additions to the ISO / IEC 42010 standard within the scope of this thesis.

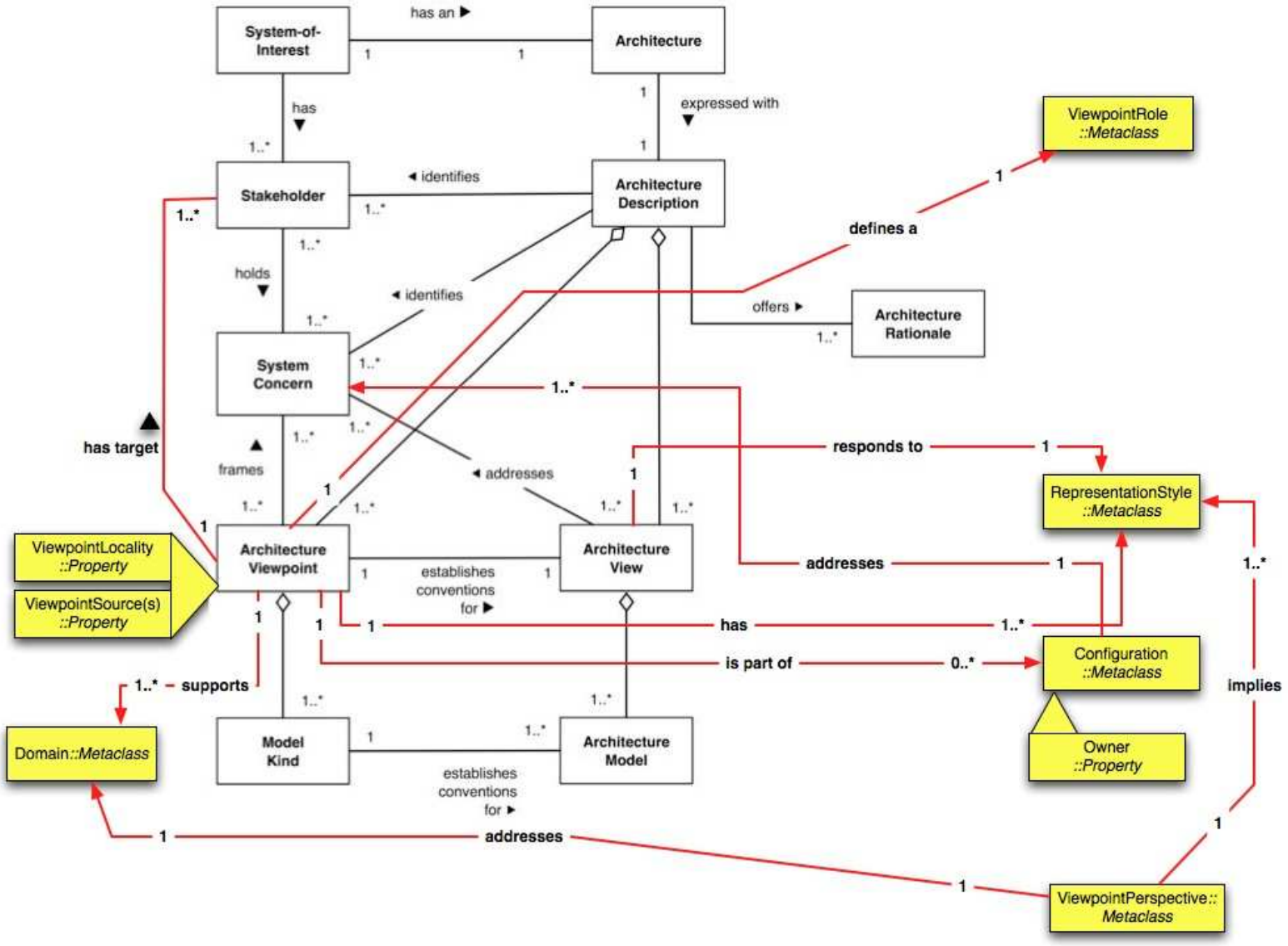

FIGURE 5.5: The ISO / IEC Core Model augmented with our own additions (in red and yellow).

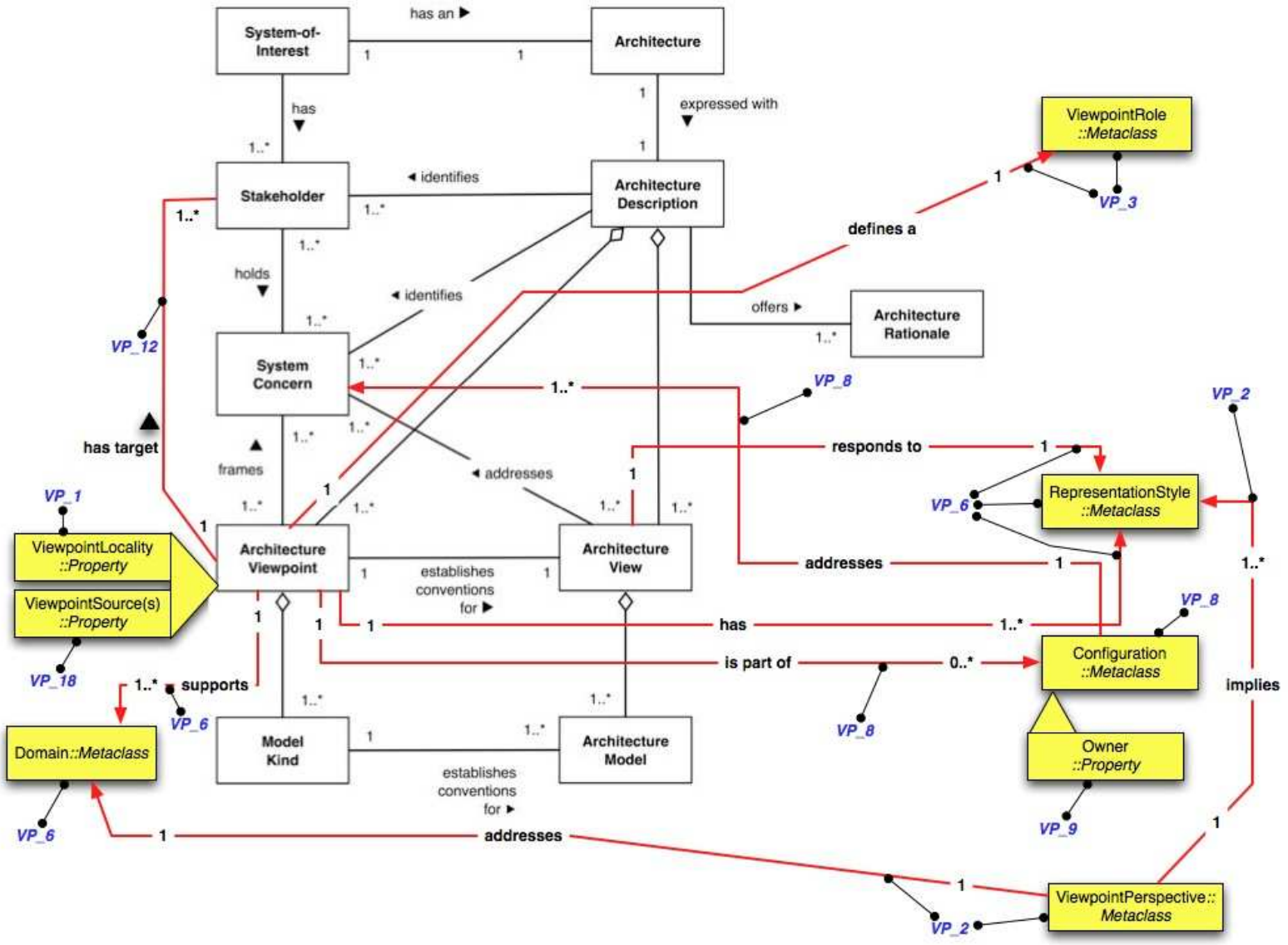

FIGURE 5.6: The ISO / IEC Core Model with augmentations and requirement tracing (in blue).

5.2 VPWiki Implementation

This section contains the refinement of the prototype architecture and the actual implementation of the viewpoint ontology in ORM through DOGMAmodeler. A high level architecture following the MVC pattern is shown in Figure 5.7.

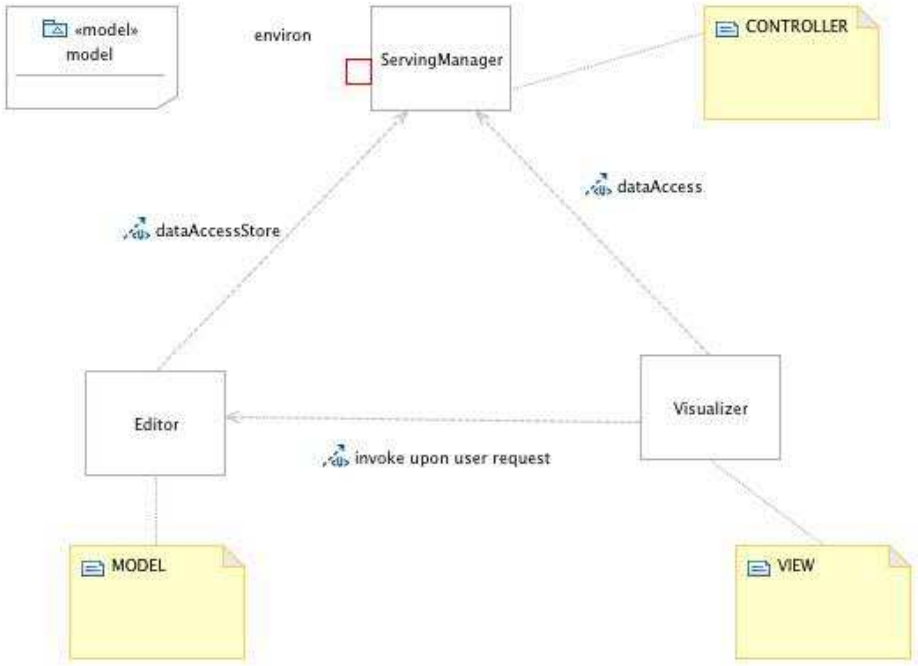

FIGURE 5.7: VPWiki: High Level Architecture - MVC Pattern.

The view in Figure 5.7 summarizes the prototype architecture at a higher abstraction level and provides evidence of its implementation pattern.

5.2.1 VPWiki Implementation: from Architecture to Classes

Figure 5.8 provides a class diagram from the components in the previous section. The component "Editor" is realized in all the green classes. The component "Visualizer" is realized in all the yellow classes. The "ServingManager" component and the persistence layer are realized partially through the yellow classes (viewing and controlling) and through external utilities (the underlying DB, the DBMS and the SPARQL query engine).

- *OntoWiki::Bootstrap*::*Interface* - This interface is used to connect to the Wiki from the external environment using the Client Server paradigm.
- *OntoWiki::ApplicationController*::*Class* - The application controller is implemented within the OntoWiki system in PHP5. It uses XAMPP to handle the persistence

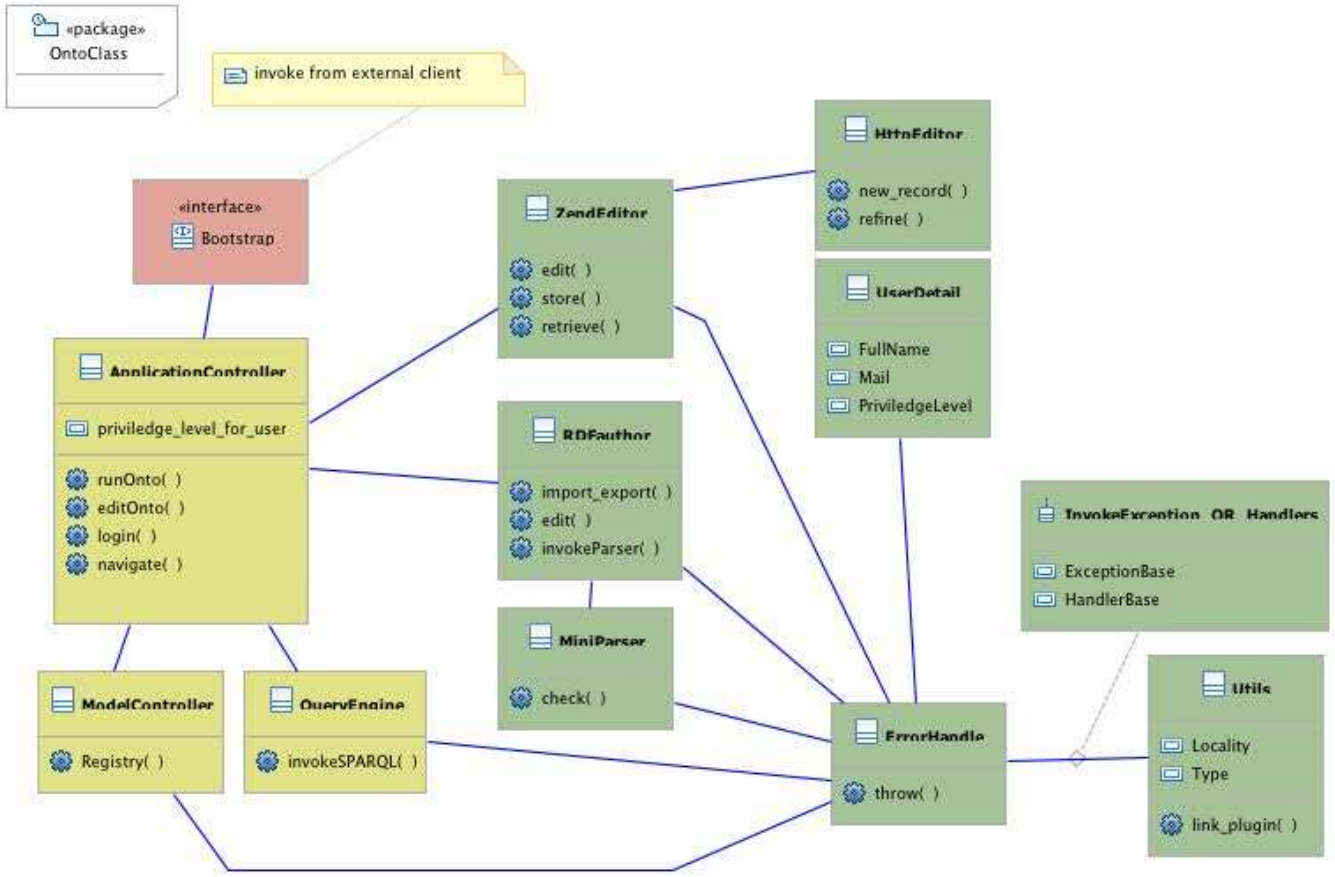

FIGURE 5.8: VPWiki: Class Diagram.

layer and is fully integrated with an HTML and RDF editor to enable editing and deployment of both records and the ontology. It is also the component charged with instantiating the querying component of the technology.

- *OntoWiki::ModelController*::*Class* - This class is implemented within the OntoWiki system in PHP5. It is used to check all the data recorded within the system either against the deployed consistency constraints or the Wiki Ontology.

- *OntoWiki::QueryEngine*::Extern::*Class* - This class invokes the query engine SPARQL present within OntoWiki. It is capable of running semantics queries as well as regular ones.

- *OntoWiki::ZendEditor*::*Class* - This editor is an instantiation of the ZEND PHP5 suite. It allows configuration and editing of the OntoWiki system itself.

- *OntoWiki::RDFauthor*::*Class* - This class is responsible for editing and redeploying the underlying ontology.

- *OntoWiki::MiniParser*::*Class* - This class re-interprets the OntoWiki system and its ontology to ensure its correct behavior. It is also invoked by the RDF author after each ontology modification.
- *OntoWiki::HttpEditor*::*Class* - This class allows editing of the public HTTP properties of the OntoWiki system.
- *OntoWiki::UserDetail*::*Class* - This class is used to define and handle user details. It is lined with the persistence layer in order to store every inserted detail and allow for further refinement.
- *OntoWiki::ErrorHandle*::*Class* - This class is invoked to offer error handling to every portion of the system. It uses external definitions and extensible mechanisms to refine both the errors and the handlers to be used within the system.
- *OntoWiki::Utils*::Extern::*Class* - This class is used by the error handler to extend the system with further definitions of error types, handlers as well as plug-ins.
- *OntoWiki::InvokeExceptions_OR_Handlers*::*AssociationClass* - This association class dynamically defines the association between the Handler and the external utilities every time a request is submitted.

All the components and classes in the architecture are already present and implemented within the OntoWiki system. All that is needed is to modify, re-configure and re-deploy OntoWiki, essentially transforming it in VPWiki.

The View and Controller classes need sensible modification since they are needed to support Viewpoints. Here follows a list of the modifications to be applied:

- Model Components:
  1. The model side is tailored so as to enforce all the constraints present in the ontology while inhibiting the possibility of inserting content other than architecture view and viewpoint specific material. This is achieved by invoking a consistency check every time some content is inserted in the system.
  2. The model side is integrated with mechanisms that handle compatibility with legacy formats.
  3. the model side is locked with the viewpoint ontology so that only view and viewpoint related materials are maintained.
- View Components:

1. The visualization components are tailored so that only a logged-in viewer can modify or insert content. This is because VPWiki needs to keep track of insertions and modifications of records.
2. The visualization components also inhibit the exploration of the records and restricts it to logged-in users.

- Controller Components:

1. The controller element is tailored so that it periodically explores the web looking for valuable additions. It points these additions out in the homepage of the admin as a modification item tagged with "upgrade".
2. The controller element notifies the users if new content is added.

5.2.2 Ontology Implementation

This section describes the implementation in Ontolanguage ORM [1], of the Ontology previously described in section 5.1.

The formalization of the Viewpoint Ontology takes place within the DOGMAmodeller technology and following the DOGMA ontological enrichment approach introduced and chosen in chapter 4.

More information of DOGMAmodeller and how it upholds the DOGMA ontology development guidelines, can be found at *http://www.jarrar.info/Dogmamodeler/*.

The process of developing and enriching our Ontology is articulated in three phases:

1. *Design the Ontology Base within DOGMAmodeller*: DOGMAmodeller is a model-driven tool, designing the ontology within it implies reproducing the meta-model.
2. *Adapt the ORM format to RDF, to import within OntoWiki*: DOGMAmodeller is provided with an import / export plug-in in different formats, exporting a design in RDF is indeed possible.
3. Import it within OntoWiki to build VPWiki: OntoWiki is provided with an import / export plug-in which can process RDF files, after appropriate configuration.

[1] http:// www.orm.net/

Chapter 6

Conclusion and Future Work

This thesis presented the design and implementation of a software architecture description viewpoint semantic wiki. Since this technology required two elements to come together, namely a *semantic wiki engine* and a meaningful *ontology*, the design started by eliciting requirements from literature for both semantic wikis and ontologies.

These requirements have subsequently been used to evaluate both semantic wiki engines and ontology engineering practices, in order to select a best-fit from both worlds. Implementation consisted in tailoring the chosen semantic wiki engine as well as developing the ontology through the selected engineering practice, respectively.

We argue that together, these two elements provide a semantic wiki to tackle the problem stated in section 1.2: **VPWiki**.

Future work on this technology focuses on integrating a number of improvements:

(i) The technology itself should be validated by skilled domain experts. Such a validation process can make sure that the technology itself is effectively meaningful. Gathering feedback is also important to improve the functional characteristics of the technology itself. A black-box evaluation of the technology by a focus group is a valuable way to carry out such an attempt.

(ii) The technology should be provided with mechanisms that allow easy extension and integration with third party technologies. This can be achieved either by developing ad-hoc APIs which can be used by third party developers to access VPWiki or by opening the default OntoWiki extension points. This second alternative will imply a more accurate security mechanism to be on place, since OntoWiki's default extension mechanisms do not regulate access and modification policies.

(iii) The technology should be provided with an ad-hoc crawling technology which periodically advises if new content is available. It would be useful if such a technology could selectively crawl within specific sites (e.g. ISO / IEC's Home Site, IEEExplore, Bibsonomy etc.) looking for advancements in viewpoints' technologies, definitions or success-stories.

(iv) The technology might also be provided with a mechanism allowing off-line integration with ADL tool-suites so as to allow recognition of viewpoints within an existing design. This mechanism can be realized with model-matching technologies [67] as well as model-to-model transformation technologies. This second technology can be used to re-design an existing model in accordance to a given view while generating a difference model between the origin model and the newly designed one [1].

[1] http://www.eclipse.org/gmt/amw/usecases/diff/

Bibliography

[1] Mark W. Maier, David Emery, and Rich Hilliard. Ansi / ieee 1471 and systems engineering. *Syst. Eng.*, 7(3), 2004.

[2] ISO / EIC consortium. Iso / iec cd 0.8 42010, 2010. http://www.iso.org/iso/home.htm.

[3] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, and Judith Stafford. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, Boston, MA, 2003.

[4] Nick Rozanski and Eowin Woods. *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley Professional, 25th April 2005.

[5] Alexander Egyed and Rich Hilliard. Architectural integration and evolution in a model world. In *Proceedings Fourth International Software Architecture Workshop (ISAW-4), 4 and 5 June 2000*, pages 37–40, 2000.

[6]

[7] Philippe Kruchten. Architectural blueprints: The "4+1" view model of software architecture. *IEEE Software*, 12(6):42–50, November 1995.

[8] Nick Rozanski and Eowin Woods. Applying viewpoints and views to software architecture. *http://www.viewpoints-and-perspectives.info*, 2004.

[9] R. N. Charette. Why software fails [software failure]. *IEEE Spectrum*, 42(9):42–49, September 2005. ISSN 0018-9235. doi: 10.1109/MSPEC.2005.1502528. URL http://dx.doi.org/10.1109/MSPEC.2005.1502528.

[10] Rich Hilliard and Tim Rice. Expressiveness in architecture description languages. In *ISAW '98: Proceedings of the third international workshop on Software architecture*, New York, USA, 1998. ACM.

[11] Ivano Malavolta, Henry Muccini, Patrizio Pelliccione, and Damien Andrew Tamburri. Providing architectural languages and tools interoperability through model transformation technologies. *IEEE Transactions on Software Engineering*, 36:119–140, 2010. ISSN 0098-5589. doi: http://doi.ieeecomputersociety.org/10.1109/TSE.2009.51.

[12] George F. Hurlburt. Development of the warfighting architecture requirements (war) tool. In *WORDS*, pages 97–104. IEEE Computer Society, 2005. ISBN 0-7695-2347-1. URL http://dblp.uni-trier.de/db/conf/words/words2005.html#Hurlburt05.

[13] Kerry Raymond. Reference model of open distributed processing - introduction. pages 3–14, 1995.

[14] Trosky Boris Callo Arias, Pierre America, and Paris Avgeriou. Defining execution viewpoints for a large and complex software-intensive system. In *WICSA/ECSA*, pages 1–10. IEEE, 2009. URL http://dblp.uni-trier.de/db/conf/wicsa/wicsa2009.html#AriasAA09.

[15] Ammar Bessam and Mohamed Tahar Kimour. Software architecture behavior metamodel for real-time systems. *Dependability of Computer Systems, International Conference on*, 0:245–252, 2008. doi: http://doi.ieeecomputersociety.org/10.1109/DepCoS-RELCOMEX.2008.12.

[16] Nenad Medvidovic, Eric M. Dashofy, and Richard N. Taylor. Moving architectural description from under the technology lamppost. *Information and Software Technology*, 49(1):12–31, 2007.

[17] Neno Medvidovic. A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*, 26:70–93, 1996.

[18] OpenGroup Standards. Architecture description languages: An overview, 1999. www.opengroup.org/architecture/ togaf/bbs/9910wash/adl_over.pdf.

[19] Lidia Rovan. In Christian Bizer and Anupam Joshi, editors, *International Semantic Web Conference*.

[20] Rik Farenhorst and Hans van Vliet. Experiences with a wiki to support architectural knowledge sharing. 2008. URL http://doc-it.fe.up.pt/wikis4se/Experiences_with_a_Wiki_to_Support_Architectural_Knowledge_Sharing.

[21] Michael C. Daconta, Leo J. Obrst, and Kevin T. Smith. *The Semantic Web: A Guide to the Future of XML, Web Services and Knowledge Management*. Wiley, Indianapolis, IN, 2003. ISBN 978-0-471-43257-9.

[22] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006. URL http://dblp.uni-trier.de/db/journals/expert/expert21.html#ShadboltBH06.

[23] Chi-Yen Yin, Yau-Jung Lee, and Jiann-Min Yang. Ontology: the historical review and literature productivity analysis using bibliometric methodology from 1956 to 2008. In *ICIS '09: Proceedings of the 2nd International Conference on Interaction Sciences*, pages 1346–1350, New York, NY, USA, 2009. ACM. doi: http://doi.acm.org/10.1145/1655925.1656172.

[24] Holger Knublauch. Ontology-driven software development in the context of the semantic web: An example scenario with protege/owl. In David S. Frankel, Elisa F. Kendall, and Deborah L. McGuinness, editors, *1st International Workshop on the Model-Driven Semantic Web (MDSW2004)*, 2004.

[25] Qing Gu and Patricia Lago. On service-oriented architectural concerns and viewpoints. In *WICSA/ECSA*, pages 289–292, 2009.

[26] Alistair G. Sutcliffe, Brian Gault, and Neil A. M. Maiden. Isre: immersive scenario-based requirements engineering with virtual prototypes. *Requir. Eng.*, 10(2):95–111, 2005.

[27] John W. Creswell. *Qualitative inquiry and research design : choosing among five traditions / John W. Creswell.* Sage Publications, Thousand Oaks, Calif. :, 1998.

[28] Dr. Robert V. Labaree. *Selected Bibliography on Qualitative Research Design.* University of Southern California.

[29] Maryam Razavian and Patricia Lago. A Frame of Reference for SOA Migration. 2010.

[30] S. Auer, S. Dietzold, and T. Riechert. Ontowiki Ð a tool for social, semantic collaboration. In Cruz, editor, *ISWC 2006, 5th International Semantic Web Conference, Nov 5th-9th, Athens, GA, USA*, pages 736–749, Berlin, 2006. Springer-Verlag. URL http://www.informatik.uni-leipzig.de/~{}auer/publication/ontowiki.pdf.

[31] Dimitris Panagiotou and Gregoris Mentzas. A comparison of semantic wiki engines. In *22nd European Conf. on Operational Research*, 2007.

[32] R. Tazzoli, P. Castagna, and S.E. Campanini. Towards a semantic wiki wiki web. In *Proceedings of the International Semantic Web Conferenc (ISWC)*, 2004.

[33] M. Schatten, M. Oubrilo, and J. Seva. A Semantic Wiki System Based on F-Logic. In *Central European Conference on Information and Intelligent Systems 19 th*

International Conference 2008 Proceedings. Faculty of Organization and Informatics, Pavlinska 2, Varazadin, 42000, Croatia, 2008.

[34] S. Schaffert, R. Westenthaler, and A. Gruber. IkeWiki: A user-friendly semantic wiki. In *3rd European Semantic Web Conference (ESWC06)*, 2006.

[35] Denny Vrandecic and Markus KrŽtzsch. Reusing ontological background knowledge in semantic wikis. In Max VŽlkel and Sebastian Schaffert, editors, *SemWiki*, volume 206 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006. URL http://dblp.uni-trier.de/db/conf/semwiki/semwiki2006.html#VrandecicK06.

[36] A. Finkelstein, J. Kramer, and J. K. Goedicke. Viewpoints oriented software specification. In *3rd International Workshop on Software Engineering and its Applications*, pages 337–351, Toulouse, France, 1990. IEEE Computer Society.

[37] B. Nuseibeh, J. Kramer, and A. Finkelstein. A framework for expressing the relationships between multiple views in requirements specification. *IEEE Transactions on Software Engineering*, 20(10):760–773, 1994. ISSN 0098-5589. doi: 10.1109/32.328995.

[38] Anthony Finkelstein and Ian Sommerville. The viewpoints FAQ. *Software Engineering Journal: Special Issue on Viewpoints for Software Engineering*, 11(1):2–4, 1996.

[39] R. Hilliard. Health-watcher: Architecture description. *ANSI/IEEE Std 1471 :: ISO/IEC 42010 web site reference*.

[40] Rallou Thomopoulos. Expressing preferences in a viewpoint ontology. In Robert Meersman, Zahir Tari, Mohand-Said Hacid, John Mylopoulos, Barbara Pernici, Ězalp Babaoglu, Hans-Arno Jacobsen, Joseph P. Loyall, Michael Kifer, and Stefano Spaccapietra, editors, *OTM Conferences (2)*, volume 3761 of *Lecture Notes in Computer Science*, pages 1596–1604. Springer, 2005. ISBN 3-540-29738-3. URL http://dblp.uni-trier.de/db/conf/otm/otm2005-2.html#Thomopoulos05.

[41] M. Uschold, M. King, S. Moralee, and Y. Zorgios. The enterprise ontology. *The Knowledge Engineering Review*, 13(01):31–89, 1998. Cambridge Univ Press.

[42] Mustafa Jarrar and Robert Meersman. Formal ontology engineering in the dogma approach. In Robert Meersman and Zahir Tari, editors, *CoopIS/DOA/ODBASE*, volume 2519 of *Lecture Notes in Computer Science*, pages 1238–1254. Springer, 2002. ISBN 3-540-00106-9. URL http://dblp.uni-trier.de/db/conf/coopis/coopis2002.html#JarrarM02.

[43] Frank Hopfgartner, Thierry Urruty, Pablo Lopez, Robert Villa, and Joemon Jose. Simulated evaluation of faceted browsing based on feature selection. *Multimedia Tools and Applications*, 47(3):631–662, 2010. ISSN 1380-7501. doi: 10.1007/s11042-009-0340-6.

[44] Benjamin Adrian. Incorporating ontological background knowledge into information extraction. online, October 2009. URL http://data.semanticweb.org/papers/iswc/2009/dc/paper111.pdf.

[45] Jian Qin and Naybell HernĞndez. Ontological representation of learning objects: building interoperable vocabulary and structures. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 348–349, New York, NY, USA, 2004. ACM. ISBN 1-58113-912-8. doi: http://doi.acm.org/10.1145/1013367.1013469.

[46] Eero Hyvönen and Eetu Mäkelä. Semantic autocompletion. In *In Proceedings of the 1st Asian Semantic Web Conference (ASWC 2006), Beijing, China*, pages 739–751, 2006. URL http://dx.doi.org/10.1007/11836025_72.

[47] Sebastian Schaffert. Ikewiki: A semantic wiki for collaborative knowledge management. In *In 1st International Workshop on Semantic Technologies in Collaborative Applications (STICAÕ06*, 2006.

[48] Malte Kiesel and Dfki Gmbh. Kaukolu: Hub of the semantic corporate intranet. In *In VŽlkel et*, page 29, 2006.

[49] Lyndon J. B. Nixon and Elena Paslaru Bontas Simperl. E.p.b.: Makna and multimakna: towards semantic and multimedia capability in wikis for the emerging web. In *In: Proc. Semantics*, 2006.

[50] A. Souzis. Building a semantic wiki. *Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications]*, 20(5):87–91, 2005. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1512004.

[51] Markus KrŽtzsch, Denny Vrande?i?, and Max VŽlkel. Semantic MediaWiki. In *The Semantic Web - ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*, pages 935–942, Heidelberg, DE, 2006. Springer. URL http://dx.doi.org/10.1007/11926078_68.

[52] E. Oren. SemperWiki: a semantic personal Wiki. In *Proc. of 1st WS on The Semantic Desktop, Galway, Ireland*, 2005.

[53] A. El Ghali, A. Tifous, M. Buffa, A. Giboin, and R. Dieng-Kuntz. Using a semantic wiki in communities of practice. In *2nd Intern. Workshop on Building Technology Enhanced Learning Solutions for Communities of Practice*, 2007.

[54] Gayo Diallo, Khaled Khelif, Olivier Corby, Patty Kostkova, and Gemma Madle. Semantic browsing of a domain specific resources: The corese-neli framework. In *Web Intelligence/IAT Workshops*, pages 50–54. IEEE, 2008. URL http://dblp.uni-trier.de/db/conf/iat/iatw2008.html#DialloKCKM08.

[55] Bo Hu and Bin Hu. On capturing semantics in ontology mapping. *World Wide Web*, 11(3):361–385, 2008. ISSN 1386-145X. doi: 10.1007/s11280-008-0042-4.

[56] Helena Sofia Pinto, Steffen Staab, and Cristoph Tempich. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, Valencia, Spain, 2004.

[57] York Sure, Rudi Studer, Contactperson Dieter Fensel, Schweizerische Lebensversicherungsund, Contactperson Ulrich Reimer, Contactperson Rudi Studer, and Martlesham Heath. On-to-knowledge methodology - expanded version, 1999.

[58] M. Fernández, A. Gómez-Pérez, and N. Juristo. Methontology: from ontological art towards ontological engineering. In *Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering*, 1997.

[59] Natalya F. Noy and Deborah L. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical report, Stanford Knowledge Systems Laboratory and Stanford Medical Informatics, 2001.

[60] M. Uschold and M. Gruninger. Ontologies principles, methods and applications. *Knowledge Sharing and Review*, 11(2), June 1996.

[61] Konstantinos Kotis and A. Vouros. Human-centered ontology engineering: The hcome methodology. *Knowl. Inf. Syst.*, 10(1):109–131, 2006. ISSN 0219-1377. doi: http://dx.doi.org/10.1007/s10115-005-0227-4.

[62] Antonio De Nicola, Michele Missikoff, and Roberto Navigli. A proposal for a unified process for ontology building: Upon. In Kim Viborg Andersen, John K. Debenham, and Roland Wagner, editors, *DEXA*, volume 3588 of *Lecture Notes in Computer Science*, pages 655–664. Springer, 2005. ISBN 3-540-28566-0. URL http://dblp.uni-trier.de/db/conf/dexa/dexa2005.html#NicolaMN05.

[63] M.B. Sarder, S. Ferreira, J. Rogers, and D.H. Liles. A methodology for design ontology modeling. In *Management of Engineering and Technology, Portland International Center for*, pages 1011–1018, 2007. ISBN 9781890843151. doi: 10.1109/PICMET.2007.4349422. URL http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4349422.

[64] JŐrŹme Euzenat and Jrme Euzenat. A protocol for building consensual and consistent repositories, 1997.

[65] Alexander GarcŠa Castro, Philippe Rocca-Serra, Robert Stevens, Chris F. Taylor, Karim Nashar, Mark A. Ragan, and Susanna-Assunta Sansone. The use of concept maps during knowledge elicitation in ontology development processes - the nutrigenomics use case. *BMC Bioinformatics*, 7:267, 2006. URL http://dblp.uni-trier.de/db/journals/bmcbi/bmcbi7.html#CastroRSTNRS06.

[66] VinŠcius Rodrigues UzŘda, Thiago Alexandre Salgueiro Pardo, and Maria das GraƊas Volpe Nunes. Evaluation of automatic text summarization methods based on rhetorical structure theory. In Jeng-Shyang Pan, Ajith Abraham, and Chin-Chen Chang, editors, *ISDA (2)*, pages 389–394. IEEE Computer Society, 2008. ISBN 978-0-7695-3382-7. URL http://dblp.uni-trier.de/db/conf/isda/isda2008-2.html#UzedaPN08.

[67] Konrad Voigt, Petko Ivanov, and Andreas Rummler. Matchbox: combined meta-model matching for semi-automatic mapping generation. In Sung Y. Shin, Sascha Ossowski, Michael Schumacher, Mathew J. Palakal, and Chih-Cheng Hung, editors, *SAC*, pages 2281–2288. ACM, 2010. ISBN 978-1-60558-639-7. URL http://dblp.uni-trier.de/db/conf/sac/sac2010.html#VoigtIR10.